Quantum Computing 4/12/2022

Lecture 20: Finding a Hidden Subgroup

Instructor: Dieter van Melkebeek Scribe: Austen Fan

The lecture today is on the Hidden Subgroup Problem (HSP). The HSP is a generalization of the problem of finding a hidden XOR-shift. It makes use of the Fourier transform over more general groups than the ones we have already considered.

First, we go over the solution to the exercise from the last lecture. Second, we discuss the quantum Fourier transform which differs from the classical Fourier transform by acting on quantum superpositions. There are some interesting situations where the quantum Fourier transform plays a critical role, such as integer factorization, which is discussed later on in the course. Next, we introduce the HSP in its generality, and present a number of its instantiations. We see that pretty much every elementary problem that we have discussed so far, for which we came up with an efficient algorithm, can be viewed as an instantiation of the HSP. Then we discuss quantum algorithms for this HSP, in particular for finite Abelian groups.

1 Solution to Last Lecture's Exercise

Exercise 1. *Consider functions $f : G \to \mathbb{C}$ that can be written as a linear combination of the characters χ:*

$$f = \frac{1}{\sqrt{|G|}} \sum_{\chi} \hat{f}(\chi)\overline{\chi} \tag{1}$$

(a) Show that if f and g can be written in the form (1), then so can $f*g : x \mapsto \sum_{y\in G} f(y)g(x-y)$.

(b) Show that $\widehat{f*g}(\chi) = c(G)\cdot \hat{f}(\chi)\cdot\hat{g}(\chi)$.

(c) Determine $c(G)$.

Solution We start with the convolution of f and g at x which is equal to the sum over all y's of $f(y)$ times $g(z)$ where $z = x - y$

$$(f*g)(x) \doteq \sum_{y\in G} f(y)g(x-y).$$

By rewriting f and g as a linear combination of the characters, the RHS becomes

$$\frac{1}{|G|}\sum_{y\in G}\left(\sum_{\chi_1}\hat{f}(\chi_1)\cdot\overline{\chi_1}(y)\right)\cdot\left(\sum_{\chi_2}\hat{g}(\chi_2)\cdot\overline{\chi_2(x-y)}\right).$$

Note that we introduce a normalization factor of $\frac{1}{\sqrt{G}}$ for both terms and combine them to $\frac{1}{|G|}$. Next, we can rewrite $\chi_2(x-y)$ using the properties of characters. Namely, that the character of a sum equals the product of the character values. Additionally, applying a character to $-y$ is the same as taking the inverse, but since all characters are on the unit sphere, taking the inverse is the same as taking the complex conjugate. So, $\chi_2(x-y) = \chi_2(x) \cdot \overline{\chi_2(y)}$ and taking the complex conjugate of the whole thing leaves us with:

$$\frac{1}{|G|} \sum_{y \in G} \left(\sum_{\chi_1} \hat{f}(\chi_1) \cdot \overline{\chi_1}(y) \right) \cdot \left(\sum_{\chi_2} \hat{g}(\chi_2) \cdot \overline{\chi_2(x)} \cdot \chi_2(y) \right).$$

Then after rearranging terms:

$$\frac{1}{|G|} \sum_{\chi_1, \chi_2} \hat{f}(\chi_1) \cdot \hat{g}(\chi_2) \cdot \overline{\chi_2(x)} \cdot \sum_{y \in G} \chi_2(y) \cdot \overline{\chi_1}(y).$$

We then notice that $\sum_{y \in G} \chi_2(y) \cdot \overline{\chi_1}(y)$ is exactly our definition of the inner product:

$$= \frac{1}{|G|} \sum_{\chi_1, \chi_2} \hat{f}(\chi_1) \cdot \hat{g}(\chi_2) \cdot \overline{\chi_2(x)} \cdot (\chi_2, \chi_1).$$

Finally, we notice that whenever χ_2 differs from χ_1 then $(\chi_2, \chi_1) = 0$, therefore the sum is only nonzero when $\chi_2 = \chi_1$ and we can then simply sum over χ. Furthermore, when $\chi_1 = \chi_2$ then $(\chi_2, \chi_1) = |G|$ which cancels with $\frac{1}{|G|}$ and all that's left is to rewrite the answer to match the given form.

$$\sum_{\chi} \hat{f}(\chi) \cdot \hat{g}(\chi) \cdot \overline{\chi(x)} \doteq \frac{1}{\sqrt{|G|}} \sum_{\chi} \widehat{f * g}(\chi) \cdot \overline{\chi(x)}$$

so $\widehat{f * g}(\chi) = c(G) \cdot \hat{f}(\chi) \cdot \hat{g}(\chi)$ where $c(G) = \sqrt{|G|}$.

2 Quantum Fourier Transform

Fourier Transform for a group G We defined the Fourier transform last lecture as a transformation on $f : G \mapsto \mathbb{C}$ that is linear, unitary, and turns convolutions into point-wise products. Such a transformation exists for many G, including all finite G and $(\mathbb{R}, +)$. In the case of finite Abelian groups, the transformation exists and is unique up to a permutation of the components; the normalized characters χ of G form the Fourier basis.

Quantum Fourier Transform for a group G The classical Fourier transform is applied to a vector with one component for every element of G and outputs the same number of components. However, the quantum Fourier transform is instead applied to a superposition on $\log |G|$ qubits. The quantum subroutine transforms input $\sum_{x \in G} \alpha(x)|x\rangle$ into output $\sum_{x \in G} \hat{\alpha}(x)|x\rangle$, where $\hat{\alpha}$ is the Fourier transform of α. These transformations can be realized by unitary circuits of size poly $\log |G|$ for every finite Abelian G (and some other groups). This has applications to things like integer factorization. For the special case of $\mathbb{Z}_2^n$ under addition $(+)$, we already know the way to realize the quantum Fourier transform, namely to apply $H^{\otimes n}$. For the other special case of $\mathbb{Z}_N$, under addition $(+)$, we will see next lecture.

3 Hidden Subgroup Problem – Statement

The HSP is a generalization of finding a hidden XOR-shift as we have discussed before. We start with formalizing the computational problem.

Input A blackbox $f : G \to R$ for some group G and set R such that for some subgroup H of G:

$$f(x_1) = f(x_2) \Leftrightarrow Hx_1 = Hx_2$$

where $Hx \doteq \{h \cdot x : h \in H\}$ is the right coset of x modulo H, and $\cdot$ denotes the group operation in multiplicative notation. An equivalent statement to the one above would be:

$$f(x_1) = f(x_2) \Leftrightarrow x_1 \cdot x_2^{-1} \in H.$$

In summary, the value that our blackbox takes on an element x of the group only depends on which right coset x belongs to, and it takes distinct values on distinct cosets.

Output A set S of generators for H, i.e., $S \subseteq G$ such that $H = \langle S \rangle \doteq \{s_1 \cdot s_2 \dots s_k : s_1, s_2 \dots, s_k \in S, k \in \mathbb{N}\}$ There always exists a set of generators of a size at most $\log_2(|H|)$. Since H is a subset of G and $|G|$ can be at most exponential in the number n of qubits, there is hope for a quantum algorithm running in time polynomial in n.

4 Hidden Subgroup Problem – Instantiations

Recall that our blackbox f has the following property:

$$f(x_1) = f(x_2) \Leftrightarrow Hx_1 = Hx_2 \Leftrightarrow x_1 \cdot x_2^{-1} \in H.$$

Problems we have already seen

Constant vs balanced for $n = 1$ We are given a function f from 1 bit to 1 bit, and we want to know whether f is constant or balanced.

$$f : \{0,1\} \to \{0,1\}$$

We claim that this can be viewed as an instantiation of the Hidden Subgroup Problem. Indeed, if we consider $G = \mathbb{Z}_2$ under addition, then $H = G$ in the constant case, and $H = \{0\}$ in the balanced case.

Learning linear functions We are given a Boolean function that maps n bits to 1 bit of the form:

$$f : \{0,1\}^n \to \{0,1\} : x \mapsto a \cdot x \text{ for some } a \in \{0,1\}^n.$$

Our output is simply a. To cast this as an instantiation of the HSP we write:

$$f(x_1) = f(x_2) \Leftrightarrow a \cdot x_1 = a \cdot x_2 \Leftrightarrow a \cdot (x_1 - x_2) = 0 \Leftrightarrow x_1 - x_2 \in a^{\perp}.$$

Therefore we can cast learning linear functions as an instatiation of the HSP where the group is $G = \mathbb{Z}_2^n$ under component-wise addition and the hidden subgroup $H = a^{\perp}$. The solution to the HSP gives us a set of generators for $a^{\perp}$, but we want a, not a set of generators for $a^{\perp}$. To get a, we solve the homogeneous system linear equations of the form $a \cdot y = 0$ where y are the generators for $a^{\perp}$; the solution to these equations will give us a.

Finding a hidden XOR-shift For this hidden XOR-shift problem, we are given some function $f : \{0,1\}^n \to \{0,1\}^n$ such that for some nonzero shift $s \in \{0,1\}^n$:

$$f(x_1) = f(x_2) \Leftrightarrow x_1 = x_2 \vee x_1 = x_2 \oplus s$$

with a goal of finding s. To frame it as an instantiation of the HSP, we write:

$$f(x_1) = f(x_2) \Leftrightarrow x_1 - x_2 \in \{0, s\}.$$

The group for this problem is $G = \mathbb{Z}_2^n$ under component-wise addition, with $H = \{0, s\}$.

Problems we have not yet seen

Period finding We have a function $f : \mathbb{Z} \to \mathbb{Z}$ such that for some nonzero $p \in \mathbb{N}$:

$$f(x_1) = f(x_2) \Leftrightarrow x_1 - x_2 \text{ is a multiple of } p\text{, or equivalently, equals 0 modulo } p.$$

The goal is to output p. Specifically, for this problem we require that there are no identical values within the period p. All values must be unique. We can cast this as an instantiation of the HSP for group $G = \mathbb{Z}$ under addition, with $H = \langle p \rangle$. Integer factorization reduces to this problem of period finding, and even to the special case of finding the order of an integer a modulo another integer μ (that is relatively prime to a). So, if we can efficiently solve period finding or order finding, we know that we can efficiently factor integers. In fact this is the key ingredient in Shor's algorithm for integer factorization.

Discrete log It is a fact that for prime p, the multiplicative group:

$$\mathbb{Z}_p^\times \doteq \{x \in \mathbb{Z}_p : \gcd(x,p) = 1\} = \{1, 2, \ldots, p-1\}$$

is cyclic. Therefore, there exists a single element which can generate every element in the group. Given a prime p, a generator g for $\mathbb{Z}_p^\times$, and an arbitrary element $a \in \mathbb{Z}_p^\times$, our goal is to output the unique $\ell \in \mathbb{Z}_{p-1}$ such that $g^\ell = a$. We know this is possible because a belongs to $\mathbb{Z}_p^\times$ and that g is a generator for the cyclic group $\mathbb{Z}_p^\times$. To cast this problem as a HSP requires some ingenuity. We make use of the following function:

$$f : \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \to \mathbb{Z}_p : (x, y) \mapsto a^x g^y \bmod p.$$

We want to know when the pair (x_1, y_1) and (x_2, y_2) map to the same value under f:

$$f(x_1, y_1) = f(x_2, y_2) \Leftrightarrow a^{x_1} g^{y_1} = a^{x_2} g^{y_2} \bmod p.$$

We know that a can be written as g^ℓ so we have:

$$\Leftrightarrow g^{\ell x_1 + y_1} = g^\ell x_2 + y_2 \bmod p.$$

Since g is a generator, the two powers of g are the same if and only if the exponents are the same modulo $p-1$

$$\Leftrightarrow \ell x_1 + y_1 = \ell x_2 + y_2 \bmod (p-1).$$

Now rearranging terms:

$$\Leftrightarrow \ell(x_1 - x_2) = y_2 - y_1 \bmod (p-1).$$

Simply writing in a different format:

$$\Leftrightarrow (x_1 - x_2, y_1 - y_2) \in \langle (1, -\ell) \rangle \text{ in } (\mathbb{Z}_{p-1}^2, +).$$

Thus, the function f defines a HSP over additive group $\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$ with $H = \langle (1, -\ell) \rangle$. A solution to this HSP will give us a set of generators for H that allows us to find ℓ; any set of generators of H will allow us to efficiently retrieve ℓ.

Graph automorphism and isomorphism An automorphism is an isomorphism of an object to itself. It is a well-known fact that a permutation $\sigma \in S_n$ is an automorphism of a (simple) graph A iff the adjacency matrix of A is invariant under permuting the rows and columns by σ, where n is the number of vertices of the graph. The adjacency matrix is a square matrix (n rows, n columns) where the (i,j)-entry is 1 if the vertex i and j are adjacent, and 0 otherwise. The set of automorphisms, $\mathrm{Aut}(A)$, is a group under composition. We now cast the question of determining $\mathrm{Aut}(A)$ into HSP. Consider the function:

$$\begin{aligned} f : S_n &\to \{0,1\}^{n \times n} \\ \sigma &\mapsto \sigma(A) \end{aligned}$$

where we identify the graph A with its adjecency matrix, which is fixed. We can then see when permutations map to the same value:

$$f(\sigma_1) = f(\sigma_2) \Leftrightarrow \sigma_1(A) = \sigma_2(A) \Leftrightarrow \sigma_1^{-1} \circ \sigma_2 \in \mathrm{Aut}(A).$$

Thus, finding generators for $\mathrm{Aut}(A)$ is an instantiation of the HSP over $(S_n, \circ)$. Observe that two connected graphs A_1 and A_2 are isomorphic iff the disjoint union $A \doteq A_1 \sqcup A_2$ has an automorphism that maps a vertex from A_1 to A_2, which is the case iff at least one element in any generating set of $\mathrm{Aut}(A)$ maps A_1 to A_2. Furthermore, isomorphism of arbitrary graphs reduces to isomorphism of connected graphs by adding to each of the two graphs a special vertex that is connected to all the vertices of the graph and is made unique such that any isomorphism between A_1 and A_2 needs to map the special vertex of A_1 to the special vertex of A_2. The latter can be done by adding a second new vertex to each graph that is only connected to the first special vertex.

5 Hidden Subgroup Problem – Algorithms

Finite Abelian groups If the underlying group is a finite Abelian group, then we can efficiently solve the HSP. The algorithm is essentially a generalization of the algorithm that we have seen for finding a hidden XOR-shift. This algorithm works efficiently, provided we know the decomposition of the underling group G as a direct product of cyclic groups. We develop the algorithm in the next section. We will assume for now an efficient quantum Fourier transform over additive group $\mathbb{Z}_N$, which will be developed in the next lectures.

Other groups

Integers under addition Integers under addition is a non-finite Abelian group. The HSP over this group is equivalent to period finding. With an efficient algorithm for this HSP, we can efficiently perform integer factorization, which allows us to break crypto-systems like RSA.

Symmetric group Graph isomorphism reduces to the HSP over the symmetric group. Even though we can efficiently compute the quantum Fourier transform over the symmetric group, we do not know how to efficiently solve the HSP over the symmetric group, and an efficient quantum algorithm for graph isomorphism remains open.

Dihedral group The dihedral group (symmetries of regular N-gon, rotations and reflections) is a non-Abelian group and efficiently solving the HSP in this group is also still open. If it can be done, it would break cryptography systems based on the shortest lattice vector problem.

6 Hidden Subgroup Problem – Algorithm over Finite Abelian Groups

We now develop an efficient quantum algorithm for the Hidden Subgroup Problem (HSP) over finite Abelian groups. More precisely, we establish the result for groups that are the direct product of cyclic groups.

Theorem 1. *Consider the group*

$$G = \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2} \times \cdots \times \mathbb{Z}_{N_k} \tag{2}$$

under component-wise addition, where $N_1, N_2, \ldots, N_k \in \mathbb{N}$, and suppose that the prime factorizations of the numbers N_j for $j \in [k]$ are given. There exists a quantum algorithm that solves the hidden subgroup problem over G with error bounded by ϵ, runs in time $O(\text{poly}\log(|G|/\epsilon))$, and makes $O(\log(|G|) + \log(1/\epsilon))$ queries to the black-box. If the size of the hidden subgroup H is also given, then the algorithm is exact, runs in time $O(\text{poly}\log|G|)$, and makes $O(\log(|G|/|H|))$ queries to the black-box.

Theorem 1 applies to all of the instantiations of the HSP over the additive group $\mathbb{Z}_2^n$ that we discussed: distinguishing constant and balanced functions for $n = 1$, learning linear functions, and finding a hidden XOR-shift. There is one more instantiation of HSP over finite Abelian groups that we discussed, namely the discrete log problem over $\mathbb{Z}_p$ for prime p. As the underlying group is $G = \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$ under addition, Theorem 1 yields a quantum algorithm that runs in time $O(\text{poly}\log p)$ once we apply the polynomial-time quantum algorithm for factoring integers to the integer $p - 1$. A similar combination of Theorem 1 and the algorithm for factoring integers yields an efficient algorithm for the HSP over more complicated finite Abelian groups provided we can efficiently compute an isomorphism with a product of cyclic groups (2). The existence of an isomorphism is guaranteed by the Structure Theorem for finite Abelian groups; the isomorphism may or may not be efficiently computable.

The algorithm of Theorem 1 generalizes the one we developed for finding a hidden XOR-shift. It consists of two parts:

- An exact quantum subroutine A that outputs a uniform element of $H^\perp$, where

$$H^\perp \doteq \{g \in G : (\forall\, h \in H)\, \chi_h(g) = 1\}.$$

  The subroutine consists of Fourier sampling. It hinges on an efficient algorithm for the quantum Fourier transform over the finite Abelian group G, which we already know for the case $G = \mathbb{Z}_2^n$ (the n-fold Hadamard transform), and which we will develop in full generality for groups of the form (2) in the next lectures.

- A classical part which uses the quantum subroutine A to construct a set S of generators for H. In the case of finding a hidden XOR-shift we ran A a number of times to find a set of generators of $H^\perp$, and then solved the system of linear equations they define in the components of the hidden shift s. In the general case, the process will similarly first find a set of generators for $H^\perp$, and then solve several systems of modular equations, each one yielding an element of a generating set S for H. More precisely, the process runs the quantum subroutine $O(\log|G| + \log(1/\epsilon))$ times to obtain a generating set for $H^\perp$ with probability at least $1-\epsilon$. Like in the special case of finding a hidden XOR-shift, an exact algorithm when $|H|$ is known can be obtained by amplitude amplification.

6.1 Quantum subroutine - Fourier Sampling

Let F denote the quantum Fourier transform over G. We use the Fourier transform over G because it interacts nicely with the symmetries captured by the group G. Recall the Fourier transform $f \mapsto \hat{f}$ for a finite Abelian group G is uniquely given by the following formula:

$$\hat{f}(y) = \frac{1}{\sqrt{|G|}} \sum_{x \in G} f(x)\chi_y(x)$$

where χ_y denotes the characters of G for $y \in G$. The corresponding quantum Fourier transform is the quantum subroutine which transforms the input $\sum_{x\in G} \alpha(x)\,|x\rangle$ into the output $\sum_{y\in G} \hat{\alpha}(y)\,|y\rangle$. It can be realized by unitary circuit of size poly $\log|G|$ for every finite Abelian group G (and some others).

Consider a (right) coset state $|Hg\rangle$, which is the uniform superposition of all elements of the coset Hg,

$$|Hg\rangle \doteq \frac{1}{\sqrt{|H|}} \sum_{h\in H} |hg\rangle\,.$$

The (quantum) Fourier transform of $|Hg\rangle$ is given by the following formula:

$$\begin{aligned} F\,|Hg\rangle &= \frac{1}{\sqrt{|H|}} \sum_{h\in H} \frac{1}{\sqrt{|G|}} \sum_{y\in G} \chi_y(hg)\,|y\rangle \\ &= \frac{1}{\sqrt{|H||G|}} \sum_{y\in G} \chi_y(g) \left(\sum_{h\in H} \chi_y(h)\right) |y\rangle\,. \qquad (3) \end{aligned}$$

Exercise 2. *Show that*

$$\sum_{h\in H} \chi_y(h) = \begin{cases} |H| & \text{if } y \in H^\perp \\ 0 & \text{otherwise.} \end{cases} \qquad (4)$$

Plugging in Equation (4) into Equation (3) gives us:

$$F|Hg\rangle = \sqrt{\frac{|H|}{|G|}} \sum_{y \in H^\perp} \chi_y(g)\, |y\rangle\,. \tag{5}$$

Thus, the Fourier transform of the coset state $|Hg\rangle$ yields an equally weighted superposition over $H^\perp$. In particular, if $g = 0$ we get a uniform superposition over $H^\perp$.

The quantum subroutine acts on a system with two registers, where the first register contains elements of the domain G of the black-box function $f : G \to R$, and the second register contains elements of the range R. We start with the first register in a uniform superposition over G and the second one in the basis state $|0\rangle$, i.e. the initial state

$$\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle\, |0\rangle\,.$$

By applying our blackbox f via U_f, we obtain the transformed state

$$\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle\, |f(g)\rangle = \sqrt{\frac{|H|}{|G|}} \sum_{\text{cosets}} |Hg\rangle\, |f(Hg)\rangle\,,$$

where we used the fact that $f(g)$ only depends on the coset Hg. Next, we apply the Fourier transform F to the first register, which by Equation (5) gives us the resulting quantum state

$$\sqrt{\frac{|H|}{|G|}} \sum_{\text{cosets}} F\,|Hg\rangle\, |f(Hg)\rangle = \frac{|H|}{|G|} \sum_{\text{cosets}} \sum_{y \in H^\perp} \chi_y(g)\, |y\rangle\, |f(Hg)\rangle\,.$$

As distinct cosets have distinct values under f and $|\chi_y(g)| = 1$ for every $g \in G$, measuring the first register yields a y uniformly at random from $H^\perp$.

6.2 Use of the quantum subroutine

Running the quantum subroutine $O(\log(|H^\perp|) + \log(1/\epsilon)) = O(\log(|G|) + \log(1/\epsilon))$ times and collecting all elements s' yields a generating set S' for $H^\perp$ with error bounded by ϵ. In order to construct a generating set S for H out of S', we make use of the fact that $(H^\perp)^\perp = H$. The fact can be argued as follows.

Exercise 3. *Show that*

1. *The quotient group G/H is isomorphic to $H^\perp$.*
2. $|G| = |H| \cdot |H^\perp|$.
3. $(H^\perp)^\perp = H$.

We can efficiently construct a generating set S for $(H^\perp)^\perp$ out of a generating set S' for $H^\perp$ in the following way, with bounded error. The elements $s \in (H^\perp)^\perp$ are exactly those that satisfy $\chi_{s'}(s) = 1$ for all $s' \in S'$. Recall that G is of the form (2), so we can write $s = (s_1, s_2, \ldots, s_k)$

and $s' = (s'_1, s'_2, \ldots, s'_k)$ where $s_j, s'_j \in \mathbb{Z}_{N_j}$ for each $j \in [k]$. Using the formula we derived for the characters of additive groups of the form (2), we have that

$$\chi_{s'}(s) = \prod_{j=1}^{k} \exp\left(2\pi i s'_j s_j / N_j\right) = \exp\left(2\pi i \sum_{j=1}^{k} s'_j s_j / N_j\right).$$

Thus, $\chi_{s'}(s) = 1$ iff $\sum_{j=1}^{k} s'_j s_j / N_j \in \mathbb{Z}$, which is equivalent to the integral modular equation

$$\sum_{j=1}^{k} \frac{M}{N_j} s'_j \cdot s_j = 0 \bmod M, \tag{6}$$

where $M \doteq \operatorname{lcm}(N_1, N_2, \ldots, N_k)$.

Theorem 2. *Given the prime decomposition of M, we can classically do both of the following in time* $\operatorname{poly}(n, \log M)$ *for a system of at most n linear equations in at most n variables over $\mathbb{Z}_M$:*

(a) Deterministically compute the number of solutions and, in particular, decide whether the system is solvable.

(b) If a solution exists, deterministically compute one as well as generate a solution chosen uniformly at random among all solutions.

Proof. We use the Chinese remainder theorem to independently solve the system modulo $p_j^{e_j}$ where $M = \prod_j p_j^{e_j}$ is the prime factorization of M, and combine those solutions into solutions to the original system. The total number of solutions equals the product of the solutions modulo each $p_j^{e_j}$, and a uniform solution is obtained by combining independent uniform solutions modulo each $p_j^{e_j}$.

To achieve (a) and (b) for a system of linear equations in n variables modulo p^e, we employ the following reduction:

- If there is a coefficient that is not divisible by p, say the coefficient of variable x_k in equation $\sum_{j=1}^{n} c_j x_j = b \bmod p^e$, use it to express x_k as a linear combination of the other variables:

  $$x_k = c_k^{-1}\left(b - \sum_{k \neq j=1}^{n} c_j x_j\right) \bmod p^e, \tag{7}$$

  where $(c_k)^{-1}$ denotes the inverse of c_k modulo p^e, which exists because $\gcd(c_k, p^e) = 1$, and can be computed efficiently using the extended Euclidean algorithm. Then use (7) to eliminate x_k from the system. The reduced system has one variable less, the number of solutions remains the same, and a uniform solution to the full system is obtained from a uniform solution of the reduced system by extending it via (7).

- If every coefficient and every right-hand side is divisible by p, then replace every equation $\sum_{j=1}^{n} c_j x_j = b \bmod p^e$ by the equation $\sum_{j=1}^{n} c'_j x'_j = b' \bmod p^{e-1}$, where $c_j = p \cdot c'_j$ and $b = p \cdot b'$. There is a bijective relationship between solutions x to the original system on the one hand, and solutions x' to the reduced system combined with choices $l_i \in \mathbb{Z}_p$ for each $i \in [n]$ on the other; the connection is given by $x_i = x'_i + l_i \cdot p^{e-1}$. It follows that the number of solutions to the original system equals the number of solutions to the reduced system times p^n, and a uniform solution to the original system is obtained by picking a uniform solution of the reduced system combined with independent uniform choices for $l_i \in \mathbb{Z}_p$.

- If every coefficient is divisible by p but not every right-hand side is, then the system has no solution.[1]

We apply part (b) of Theorem 2 to generate $O(\log |H| + \log(1/\epsilon))$ independent uniformly distributed samples of the solutions to the system of equations (6). With probability at least $1 - \epsilon$, the resulting set S generates $(H^\perp)^\perp = H$.

In case $|H|$ is known, we also know $|H^\perp| = |G|/|H^\perp|$ by part (b) of Exercise 3. In that case, we can make use of amplitude amplifications with known success probability to obtain, with certainty, a generating set S' for $H^\perp$ in time $\text{poly}\log|G|$ using a number of black-box queries bounded by $O(\log |H^\perp|) = O(\log(|G|/|H|))$. We construct the set S' element by element. In each step we obtain, with certainty, an element $s' \in H^\perp$ that is not in the set generated by the current S'.[2] Once we have the generating set S' for $H^\perp$, we can similarly find a generating set S for H with certainty in the stated time and query complexity. $\square$

Note that the proof of Theorem 1 uses the fact that the underlying group G is finite Abelian in two ways:

- A small number of Fourier samples contains enough information to determine generators for the hidden subgroup H of G (and H can be retrieved efficiently from the samples).

- The quantum Fourier transform over G can be computed efficiently.

The first item hinges on the homomorphic properties of the Fourier basis, and breaks down for more general groups. In particular, for the symmetric group S_n, even though the quantum Fourier transform can be computed efficiently, one needs an exponential number of queries in n in order to obtain a significant statistical distance between the distributions of positive and negative instances of graph isomorphism.

[1]This case cannot happen for the homogenous system consisting of the equations (6), but the exact algorithm uses another application of the claim, in which the right-hand sides are not all zero.

[2]This makes use of Theorem 2 with right-hand side s'_j modulo N_j.