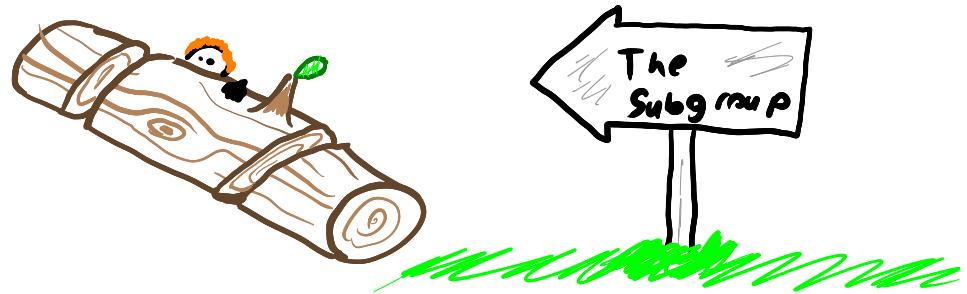
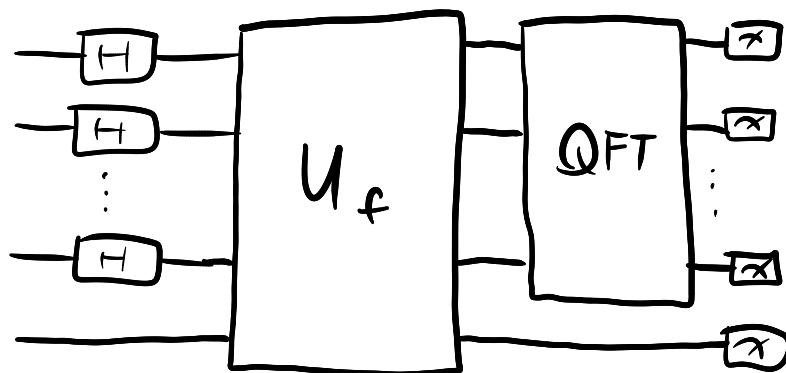


CMPT 476 Lecture 22

Discrete logarithms and the Hidden Subgroup problem



As Shor's algorithm was really just a query-efficient algorithm for finding the period of a function over \mathbb{Z}_N its generality lends itself to other applications and generalizations. Today we'll look at some applications based on generalization of the QFT to other finite groups. If you aren't familiar with groups, I would encourage looking at the group theory appendix at this point. $O(1)$ number of queries.



(Diffie-Hellman & ECC)

Both the Diffie-Hellman key exchange and elliptic-curve cryptography are based on the assumed hardness of computing discrete logarithms

$$c = \log_b a \text{ where } a, b \in G, c \in \mathbb{Z}$$

One of Shor's insights was that his period-finding algorithm **also** gives an efficient algorithm for the discrete logarithm problem in \mathbb{Z}_p^\times .

(\mathbb{Z}_p^\times - multiplicative integer group mod p)

Discrete logarithm problem in \mathbb{Z}_p^\times

formally, r is the
order of ab

The notes do not mention p being an argument to the function
Input: integers b, a , and r such that $a^r \equiv b^r \equiv 1 \pmod{p}$

Promise: $a = b^t$ for some $t \in \{0, 1, \dots, r-1\}$

$a = b^{rt} \pmod{p}$

Why is t in $[0, r-1]$?

Goal: Compute $t = \log_b a$

Anything outside the range $[0, r-1]$ becomes r and this reduces to 1.

$$b^{(r+1)t} = b^r * b^{rt} = b^r \pmod{p}$$

Problem: period of $a^x \pmod{p}$ and $b^x \pmod{p}$ is r.

We need to find the period of a and b.

Ex.

Let $p=7$, $b=3$ and $a=5$. I claim that $r=6$:

How are we using Euler's theorem?

power	1	2	3	4	5	6
a	5	$25 \equiv 4 \pmod{7}$	6	2	3	1
b	3	$9 \equiv 2 \pmod{7}$	6	4	5	1

The discrete log of a with respect to b.

$$b^5 \equiv 5 \pmod{7}$$

$$\therefore t = 5$$

In general, in \mathbb{Z}_N^\times (or any cyclic group), a basic fact of group theory states that

$$\mathbb{Z}_N^\times = \langle b \mid b \in \mathbb{Z}_N^\times \rangle = \{b^0, b^1, \dots, b^{N-1} \mid b \in \mathbb{Z}_N^\times\}$$

So it's pretty easy to come up with instances of a & b as in the problem.

(Shor's approach to DLog)

Shor's approach was to use his period-finding subroutine. However, applying it to $f(x) = a^x \bmod p$ would give the **order/period r** of a , which is independent of t !

His solution was to instead compute the period (s_1, s_2) of a two-parameter function

$$f(x, y) = a^x b^y \bmod p$$

The intuition is that the period of the above should be a function of t & r — since we know r maybe we can get t ?

$$b^{\{ts_1 + s_2\}} = 1 \bmod p$$

If f is periodic, then $f(x + s_1, y + s_2) = f(x, y)$

Note that $a^{x+s_1} b^{y+s_2} = b^{t(x+s_1)+y+s_2} = a^x b^y$ if and only if $+s_1 + s_2 \equiv 0 \pmod{r}$ (basic facts of group theory). So,

$+s_1 + s_2 \equiv 0 \pmod{r}$ Why is this a basic fact of group theory? Is it not a basic fact in general?

Why is it mod r ?

$$\begin{aligned} +s_1 + s_2 \equiv 0 \pmod{r} \Rightarrow +s_1 &\equiv -s_2 \pmod{r} \\ &+ \equiv -s_2 s_1^{-1} \pmod{r} \end{aligned}$$

Why would s_1 be coprime with r ?

At this point all we need to figure out is how to compute the two-parameter period. Recall that Simon's algorithm used $H \otimes H \otimes \dots \otimes H$ to get the period of an n -parameter function

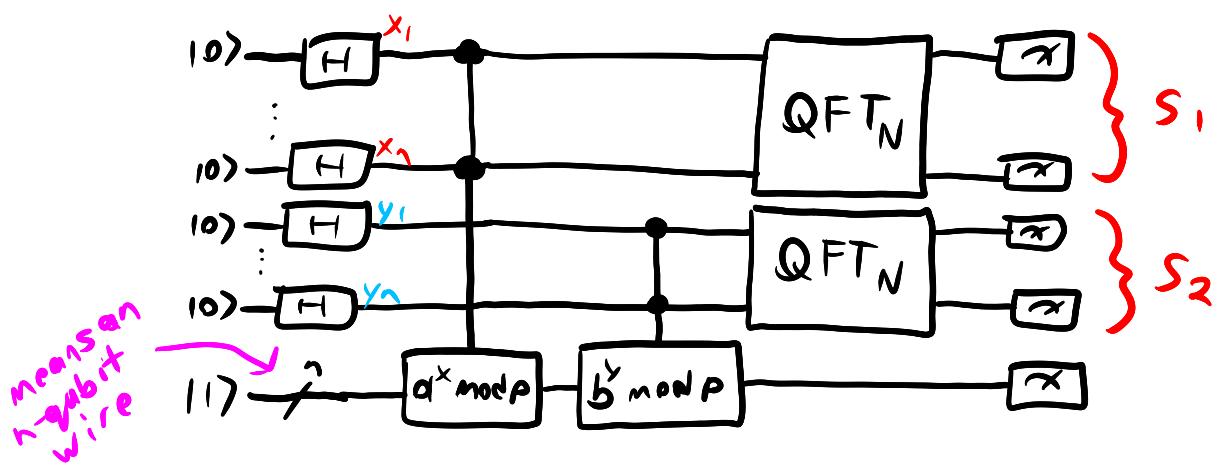
$$f: \hat{\mathbb{Z}_2} = \mathbb{Z}_2 \times \mathbb{Z}_2 \times \dots \times \mathbb{Z}_2 \rightarrow \hat{\mathbb{Z}_2}$$

where $H = QFT_Q$? Well, let's try the same thing here on our function

$$f: \hat{\mathbb{Z}_N} = \mathbb{Z}_N \times \mathbb{Z}_N \rightarrow \hat{\mathbb{Z}_N}$$

with two QFT_N 's.

(Shor's DLog algorithm)



We won't go through the whole analysis (in lieu of a more general & clean one) but it suffices to note that the periodic state generated before the parallel QFTs is

$$\sum_{x,y \in \mathbb{Z}_N} |x\rangle |y\rangle |a^x b^y \bmod p\rangle$$

For a particular $a^x b^y \bmod p = c$ we can write the sum as

$$\begin{aligned} & \sum_{x,y \in \mathbb{Z}_N} |a^x b^y \bmod p = c\rangle |x\rangle |y\rangle |c\rangle \\ &= \sum_{\substack{s_1, s_2 \in \mathbb{Z}_r \\ s_1 + s_2 \equiv 0 \pmod{r}}} |x_0 + s_1\rangle |y_0 + s_2\rangle |c\rangle \\ &= \sum_{k \in \mathbb{Z}_r} |x_0 + (-\frac{k}{r} \bmod r)\rangle |y_0 + k\rangle |c\rangle \end{aligned}$$

Where does k come from?

Taking the Fourier Transforms results in

$$\sum_{k \in \mathbb{Z}_r} \sum_{z_1, z_2} w_N^{(x_0 + (-\frac{k}{r} \bmod r))z_1 + (y_0 + k)z_2} |z_1\rangle |z_2\rangle$$

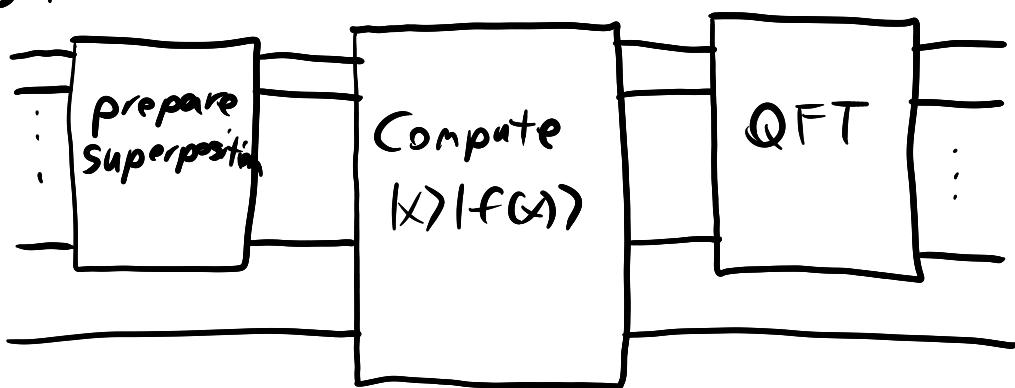
Which constructively interferes roughly when

$$-\frac{k}{r}z_1 + kz_2 \equiv 0 \pmod{N} \Rightarrow z_2 \equiv \frac{z_1}{r} \pmod{N}$$

\therefore modulo modulus issues, $z_2 \equiv z_1^{-1} \pmod{N}$

(Hidden subgroup problems)

All of the algorithms we've studied so far have the same structure:



This is because they are all solving (versions of) the same problem: the (Abelian) Hidden Subgroup Problem.

First, recall that a group $G = (S, \cdot)$ is a set S equipped with an identity $e \in S$, a binary operator $\cdot : S \times S \rightarrow S$ and inverses $a^{-1} \forall a \in S$ such that

$$1. (a \cdot b) \cdot c = a \cdot (b \cdot c)$$

$$2. a \cdot e = a = e \cdot a$$

A group is abelian if $a + b = b + a$
It allows commutative operations.

$$3. a \cdot a^{-1} = e$$

and a subgroup S of G is a subset of G that is a group with respect to G 's operations

(see appendix for more details)

Closed under inverses, binary operator.
Subgroups partitions a group into subsets.

A coset is not a subgroup of G since you modify the inverse and identity element.

The Hidden Subgroup Problem (HSP)

input: a function $f: G \rightarrow X$ where G is a group and X a finite set ($\log Z_{\log |X|}$)

promise: there exists a subgroup S of G such that

$$f(x) = f(y) \text{ if and only if } xS = yS$$

if they are the same coset.

goal: find S

What does it mean to have the x next to the S ? Am I multiplying each element of S by x ?

Ex.

Many problems of interest can be described as hidden subgroup problems. Here are a few.

If the hidden string is the zero string,

Why is s larger than 1 in general?

Is it since the subgroup contains elements which provoke collisions?

The first four problems are defined over abelian groups.

Problem	Group	f	Subgroup
Deutsch	(\mathbb{Z}_2, \oplus)	any	$S = \{0\}$ if & only iff is constant Why is this the only subgroup?
Simon	(\mathbb{Z}_2^n, \oplus)	any	$S = \{s\}$
Period finding	$(\mathbb{Z}_{N^r}, +)$	any	Once we have a period, we have all the locations where $f(x)$ outputs y . $S = r\mathbb{Z}_N$, r is order of $a \pmod{N}$
DLog	$(\mathbb{Z}_N \times \mathbb{Z}_N, +)$	$a^x b^y \pmod{p}$	$S = \mathbb{Z}_r \times (-k\mathbb{Z}_r)$, k is the Dlog $k = \log_b a$
Approx.	Dih_n (Dihedral group)		Ask professor about why this is the subgroup.
Shortest vector	S_n (Symmetric group)		These problems are <u>all</u> likely NP-intermediate (not NP-complete but not in P either) Curious...
Graph isomorphism			

The last two problems have non-abelian groups as the domain.

The last two problems above involve groups in which the binary operator is **non-commutative** — that is, they are **non-Abelian groups**. It turns out that Shor's algorithm generalizes to **all finite Abelian HSP's** thanks to the beautifully simple Fourier theory of **finite Abelian groups**, and the likewise beautiful classification of finite Abelian groups.

Breaks RSA
Period finding

Breaks ECC

Breaks
Brice lattice
basis
CFL problem
(Peg solit)

Graph isomorphism
Would be a big deal...

(Fourier theory of finite Abelian groups)

Let G be a finite Abelian group. A character of G is a homomorphism from $G \rightarrow T$ where T is the multiplicative group of roots of unity

$$T = \{z \in \mathbb{C} \mid |z|=1\}, \cdot$$

The set \widehat{G} of (irreducible) characters of G is isomorphic to G .
What are irreducible characters?

Ex.

A character of \mathbb{Z}_3 is $\chi_1: a \mapsto e^{\frac{2\pi i}{3} \cdot a}$

Observe that for any $a \equiv b \pmod{3}$, $\chi_1(a) = \chi_1(b)$.

Moreover, for any $a, b \in \mathbb{Z}_3$,

$$\chi_1(a+b) = e^{\frac{2\pi i}{3} \cdot (a+b)} = e^{\frac{2\pi i}{3} a} e^{\frac{2\pi i}{3} b} = \chi_1(a)\chi_1(b)$$

In general, \mathbb{Z}_N has characters

$$\chi_b: a \mapsto e^{\frac{2\pi i}{N} \cdot ab} \quad \forall b \in \mathbb{Z}_N$$

which satisfy $\chi_{(b+c)} = \chi_b \cdot \chi_c$, giving an isomorphism of groups $\mathbb{Z}_N \cong \widehat{\mathbb{Z}_N}$ with $b \leftrightarrow \chi_b$.
Ask Matt how this gives an isomorphism of groups.

The Fourier transform of a function $f: G \rightarrow \mathbb{C}$ is a function $\widehat{f}: \widehat{G} \rightarrow \mathbb{C}$ such that

$$f(a) = \frac{1}{|G|} \sum_{x \in G} \widehat{f}(x) \chi(a)$$

It takes a function that returns a signal and return a collection of periodic functions which sum to that signal?

Is this a way to avoid applying the Fourier transform after applying f ?

You change f so that it applies the Fourier transform?

$$\text{In particular, } \widehat{f}(x) = \sum_{a \in G} f(a) \chi(a)^*$$

Is this star the complex inverse of the element?

Ex. (normalized)

The Fourier transforms of \mathbb{Z}_2 and \mathbb{Z}_N correspond to the Hadamard gate and QFT_N , respectively:

$$f(x) = \frac{1}{\sqrt{|\mathbb{Z}_2|}} \sum_{y \in \mathbb{Z}_2} \hat{f}(y) \underbrace{x_y(x)}_{(-1)^{xy}}$$

$$f(x) = \frac{1}{\sqrt{|\mathbb{Z}_N|}} \sum_{y \in \mathbb{Z}_N} \hat{f}(y) \underbrace{x_y(x)}_{e^{2\pi i \frac{y}{N} \cdot xy}}$$

What about a group like $\mathbb{Z}_2^N = \mathbb{Z}_2 \times \mathbb{Z}_2 \times \dots \times \mathbb{Z}_2$?

Well, the characters of $G \times H$ are the product of characters of G & H , so

$$\chi_{x_1 \dots x_N}(y_1 \dots y_N) = \chi_{x_1}(y_1) \dots \chi_{x_N}(y_N)$$

Where do we get the -1 from?

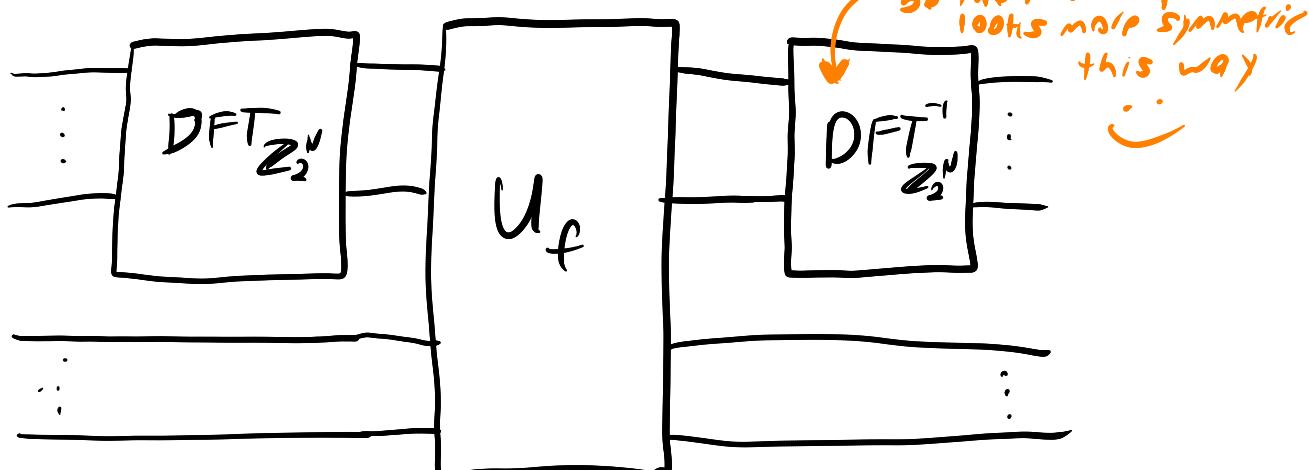
$$= (-1)^{x_1 y_1} \dots (-1)^{x_N y_N}$$

$$= (-1)^{x \cdot y} \quad \leftarrow \text{dot product in } \mathbb{Z}_2$$

So the Fourier transform on \mathbb{Z}_2^N is

$$f(x) = \frac{1}{\sqrt{|\mathbb{Z}_2^N|}} \sum_{y \in \mathbb{Z}_2^N} \hat{f}(y) (-1)^{x \cdot y}$$

But this is just $H \otimes H \otimes \dots \otimes H$! So Simon's algorithm is really just



While we presented Shor's algorithm with $H^{\otimes n}$ as the initial Fourier Transform, note that

$$QFT_N |0\rangle = \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}_N} w_N^{0y} |y\rangle = \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}_N} |y\rangle$$

so $H^{\otimes n} |0\rangle = QFT_N |0\rangle$ and Shor's algorithm does have the form above as well.

Ex.

In the discrete log algorithm, we used two QFT_N 's

$$QFT_N \otimes QFT_N$$

Like in Simon's example above, we can see that this is the Fourier transform on $G = \mathbb{Z}_N \times \mathbb{Z}_N$:

$$\begin{aligned} f(a, b) &= \frac{1}{|G|} \sum_{x \in \hat{G}} \widehat{f}(x) x(a, b) \\ &= \frac{1}{|G|} \sum_{x_1, x_2 \in \hat{\mathbb{Z}}_N} \widehat{f}(x_1, x_2) x_1(a) x_2(b) \\ &= \frac{1}{|G|} \sum_{k_1, k_2 \in \mathbb{Z}_N} \widehat{f}(k_1, k_2) w_N^{k_1 a} w_N^{k_2 b} \\ &= \frac{1}{|G|} \sum_{k_1, k_2 \in \mathbb{Z}_N} \widehat{f}(k_1, k_2) w_N^{k_1 a + k_2 b} \end{aligned}$$

Comparing with $QFT_N \otimes QFT_N$:

$$\begin{aligned} (QFT_N \otimes QFT_N)(a)(b) &= \frac{1}{\sqrt{N^2}} \sum_{k_1, k_2 \in \mathbb{Z}_N} w_N^{k_1 a} w_N^{k_2 b} |k_1, k_2\rangle \\ &= \frac{1}{\sqrt{N^2}} \sum_{k_1, k_2 \in \mathbb{Z}_N} w_N^{k_1 a + k_2 b} |k_1, k_2\rangle \end{aligned}$$

(QFT on products of \mathbb{Z}_N)

Let $G = \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2} \times \cdots \times \mathbb{Z}_{N_k}$. Then

$$QFT_G = QFT_{N_1} \otimes QFT_{N_2} \otimes \cdots \otimes QFT_{N_k}$$

(The Fourier Transform and the HSP)

Suppose we have $G = \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2} \times \dots \times \mathbb{Z}_{N_k}$ and an HSP f, S on G . Can we do the same thing we did for Simon & Shor's algorithms?

Let's do the standard steps

1. Prepare $\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle$ with QFT_G

2. Compute $\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle |f(g)\rangle$ with oracle U_f

3. Measure to get $\frac{1}{\sqrt{|C|}} \sum_g |g\rangle |c\rangle$ if $f(g) = c$

Now we know $f(g) = f(h)$ if & only if $g + S = h + S$, so we can pick a representative g_0 of the coset C and write the sum over S , giving the coset state

$$|g_0 + S\rangle = \frac{1}{\sqrt{|S|}} \sum_{s \in S} |g_0 + s\rangle$$

Applying the Fourier transform we get

$$\begin{aligned} & \frac{1}{\sqrt{|S|}} \cdot \frac{1}{\sqrt{|G|}} \sum_{s \in S} \sum_{g \in G} \chi_g(g_0 + s) |g\rangle \\ &= \frac{1}{\sqrt{|S| \cdot |G|}} \sum_{s \in S} \sum_{g \in G} \chi_g(g_0) \chi_g(s) |g\rangle \end{aligned} \quad \text{Since } \chi_g \text{ is a homomorphism}$$

Now we could show interference using the fact that $\chi_g(s)$ is a root of unity, but a simpler algebraic proof exists using properties of group characters.

Define the kernel of S as

$$S^\perp = \{t \in G \mid \chi_t(s) = 1 \text{ for all } s \in S\}$$

We claim that the final state is

$$\frac{1}{\sqrt{|S^\perp|}} \sum_{t \in S^\perp} \chi_t(g_0) |t\rangle$$

The proof relies on the fact that a character of a group is also a character of all of its subgroups, and the the orthogonality theorem

(Orthogonality theorem for characters)

Let $\chi_y, \chi_{y'}$ be characters of a group. Then

$$\sum_{g \in G} \chi_y(g) \chi_{y'}(g^*) = \begin{cases} 0 & \text{if } y \neq y' \\ |G| & \text{otherwise} \end{cases}$$

Taking $\sum_{s \in S} \chi_g(s) = \sum_{s \in S} \chi_g(s) \chi_1(s)^*$ this sum is 0

(i.e. interferes) whenever $g \notin S^\perp$ (note that $\chi_1(s) = 1$ for all $s \in S$ so $1 \in S^\perp$).

Ask professor about this point.

(An algorithm for HSP on $\mathbb{Z}_{N_1} \times \dots \times \mathbb{Z}_{N_k}$)

So, we've shown that at least when $N_i = 2^{n_i}$ for each i , we can sample from S^\perp using Simon/Shor's algorithm. In Simon's algorithm we then used samples to find a linear basis of S^\perp . Since $\mathbb{Z}_{N_1} \times \dots \times \mathbb{Z}_{N_k}$ works mostly like a vector space we can do the same

Algorithm sketch

1. Set $n \approx \sum_i \log_2 N_i$

2. Use Simon/Shor/Fourier sampling to sample

$$t_1, \dots, t_{n+4} \in S^\perp$$

With at least $\frac{2}{3}$ probability, t_1, \dots, t_{n+4} is a basis for S^\perp — in group theory terms,

$$\langle t_1, \dots, t_{n+4} \rangle = S^\perp$$

3. Solve this linear system to get a basis of S

$$\begin{bmatrix} t_1 \\ \vdots \\ t_{n+4} \end{bmatrix} \begin{bmatrix} x_1/N_1 \\ \vdots \\ x_k/N_k \end{bmatrix} = \begin{bmatrix} 0 \pmod 1 \\ \vdots \\ 0 \pmod 1 \end{bmatrix}$$

(Abelian vs non-Abelian HSP)

Thanks to the fundamental theorem of finite Abelian groups, which states that for any such group G ,

$$G \cong \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2} \times \cdots \times \mathbb{Z}_{N_k}$$

the previous algorithm solves any HSP in any finite Abelian group in poly-time. Pretty cool! 😊

The non-Abelian case is much harder, because the characters of a non-Abelian group are not simple roots of unity. Still, a poly-time algorithm for the non-Abelian case would have massive impacts via the shortest vector and graph isomorphism problems, and there's good reason to believe it should be possible.

The most famous result for the non-Abelian case is Kuperberg's sub-exponential-time algorithm for the HSP in dihedral groups. Specifically, Kuperberg's (first) algorithm runs in time

$$2^{\mathcal{O}(\sqrt{\log|G|})}$$

Compared to the best-known classical algorithm which runs in exponential time

$$\mathcal{O}(\sqrt{2^{\log|G|}})$$

A caveat of Kuperberg's algorithm however is (to the best of my knowledge) no one knows how to implement it in practice. A few years back I was at a workshop aimed at examining the real impacts of QC on crypto. The question of implementing Kuperberg's algorithm came up — with researchers much more accomplished than I we tried for a few days, before eventually giving up... I think there's still some code floating around...