Week 1 & 2: The Basics and Some Extensions

Stacey Jeffery

February 10, 2025

These notes cover: building blocks of quantum algorithms (local gates, random access gates, queries); hidden subgroup problems with applications to factoring and discrete log; phase estimation with application to search, approximate counting, and amplitude amplification.

1.1 Building Blocks

1.1.1 Problems and Algorithms

We start by describing some details about what a problem is, what an algorithm is, and what it means for an algorithm to solve a problem. You probably already have some idea of what this means, but here we review this somewhat more precisely.

Problems

(Decision) Problems A *problem* is a family of functions $\mathrm{F}_n : \{0,1\}^n \to \{0,1\}^m$, one for each $n \in \mathbb{N}$, where m is some function of n. We will often leave the "family of" part and the subscript n implicit, and just let $\mathrm{F} : \{0,1\}^n \to \{0,1\}^m$ define the problem. For example, $\mathrm{F} = \mathrm{F}_n$ might take an n-bit integer, and output a binary description of its smallest non-trivial factor. When $m = 1$, we call F a *decision problem*. For example, F might take an n-bit integer and output 1 if and only if its smallest non-trivial factor is greater than $2^{n/2-1}$.

We often restrict our attention to decision problems. These largely capture the general case, since $\mathrm{F} : \{0,1\}^n \to \{0,1\}^m$ can be described by m decision problems $\mathrm{P}^i : \{0,1\}^n \to \{0,1\}$ for $i \in [m]$, where $\mathrm{F}(x) = (\mathrm{P}^1(x), \ldots, \mathrm{P}^m(x))$. What this means concretely is that we can compute F by computing m decision problems. In fact, in quantum algorithms, we can do even better.

Exercise 1.1.1. *Fix a problem* $\mathrm{F} : \{0,1\}^n \to \{0,1\}^m$*. For any* $z \in \{0,1\}^m$*, let* $\mathrm{F}^z : \{0,1\}^n \to \{0,1\}$ *be defined* $\mathrm{F}^z(x) = z \cdot \mathrm{F}(x)$*, where* $y \cdot y' := \sum_{i=1}^m y_i y'_i$*. Suppose you can efficiently compute* F^z *for any* z*, which we will model by assuming you have query access to a unitary* $U_{\mathrm{F},x}$ *on* $\mathrm{span}\{|z\rangle : z \in \{0,1\}^m\}$ *that acts as* $U_{\mathrm{F},x} = (-1)^{\mathrm{F}^z(x)}|z\rangle$*. Show how to compute* $\mathrm{F}(x)$ *using a single call to* $U_{\mathrm{F},x}$*, and* $O(m)$ *additional gates.*

Partial Functions A *partial function* is a function (family) $\mathrm{F}_n : D_n \to \{0,1\}^m$ for some $D_n \subsetneq \{0,1\}^n$. That is, the function is only defined on a domain that is a subset of all possible strings. For example, the Deutsch-Jozsa algorithm ([DJ92] or [dW19, Chapter 2]) decides if a string $x \in \{0,1\}^n$ is *constant* or *balanced*. That is, it decides $\mathrm{DJ} : D_n \to \{0,1\}$ where D_n is the set of n-bit strings of Hamming weight (number of 1s) $|x| = 0$, $|x| = n/2$, or $|x| = n$, and

$$\mathrm{DJ}(x) = \begin{cases} 0 & \text{if } |x| \in \{0, n\} \\ 1 & \text{if } |x| = n/2. \end{cases}$$

For $|x| \notin \{0, n/2, n\}$, we do not specify the value of $\mathrm{DJ}(x)$, and any algorithm for this problem can behave arbitrarily on such inputs. Partial functions are also called *promise problems*, since we are promised that the input has a certain form. Functions that are not partial, meaning $D_n = \{0,1\}^n$ for all n, are called *total functions*.

Relations We can also consider problems where there is more than one possible correct output for a given input. Such a problem is captured by a family of *relations* (we will often call such a problem a relation) $\mathrm{R}_n \subset \{0,1\}^n \times \{0,1\}^m$, for each $n \in \mathbb{N}$, where m is some function of n. Then a "correct" output on input x is any $y \in \{0,1\}^m$ such that $(x,y) \in \mathrm{R}$. We can generally assume that for all $x \in \{0,1\}^n$ there exists at least one $y \in \{0,1\}^m$ such that $(x,y) \in \mathrm{R}$, as otherwise there is no correct output on input x, and so our algorithm cannot be expected to find one.

For example, the problem of $\textsc{Search}_n$ – finding $i \in [n]$ such that $x_i = 1$ – can be described by the relation:

$$\{(x,i) \in (\{0,1\}^n \setminus \{0^n\}) \times \{0,1\}^{\log n} : x_i = 1\} \cup \{(0^n, \perp)\}.$$

Algorithms

Quantum Algorithms A quantum algorithm $\{(U_1^n(x), \ldots, U_{T(n)}^n(x), |\psi_0^n\rangle, \mathcal{M}_n)\}_{n\in\mathbb{N}}$ consists of

1. for each $x \in \{0,1\}^n$, $U_1^n(x), \ldots, U_{T(n)}^n(x)$ is a sequence of input-dependent unitaries on some finite-dimensional complex inner product space H_n,
2. $|\psi_0^n\rangle \in H_n$ is an initial state,
3. and $\mathcal{M}_n = \{\Pi_y^n\}_{y\in\mathcal{Y}_n}$ for some finite set $\mathcal{Y}_n$ is a final measurement.

We will generally let n and/or the dependence on x be implicit unless we want to emphasize it, writing $U_1, \ldots, U_T$, for example. Running the algorithm consists of measuring $|\psi_T(x)\rangle := U_T(x) \ldots U_1(x)|\psi_0\rangle$, with $\mathcal{M}$, and outputting the measurement outcome $Y \in \mathcal{Y}$, which is a random variable distributed as $\Pr[Y = y] = \|\Pi_y|\psi_T(x)\rangle\|^2$.

The unitaries are generally restricted to be from some set of "basic operations" (otherwise they could be combined into a single unitary), and then T is some measure of the complexity of the algorithm. The specifics of this set depend on the precise model of computation. We discuss common choices for what can count as a "basic operation" in the rest of this section, but as an example, the unitaries might consist of input-independent *gates*, and input-dependent *queries* to x.

We often assume $|\psi_0\rangle = |0\rangle$ – by $|0\rangle$ we mean some kind of "neutral" or "ground" state (should be easy to prepare), which may, for example, represent the all 0s string in an M-qubit space. This assumption is reasonable, because if we actually want to start in another state, we can simply prepare it from $|0\rangle$ using the first few unitaries, or if it is actually not straightforward to prepare, then it was not a very reasonable initial state to begin with – in that case we have "hidden" a bunch of computation into preparation of the initial state.

We often assume that there is some *answer register*, meaning we can express H as $H = \mathbb{C}^{\mathcal{Y}} \otimes H'$, and that $\Pi_y = |y\rangle\langle y| \otimes I_{H'}$, where $I_{H'}$ is the identity on H'. That is, we assume the final measurement just measures the answer register. This assumption is justified by the fact that any measurement can be expressed this way up to a change of basis, and if the required change of basis is not straightforward to implement, then this was not a very reasonable final measurement – in that case we have "hidden" a bunch of computation in the implementation of the final measurement.

Bounded Error Fix $\varepsilon \in (0, 1/2)$. We say a quantum algorithm solves a problem $\mathrm{F} : D \to \{0,1\}^m$ with *bounded error* ε if for all $x \in D$,

$$\left\|\Pi_{\mathrm{F}(x)}|\psi_T(x)\rangle\right\|^2 \geq 1 - \varepsilon,$$

that is, on input x, the algorithm outputs $\mathrm{F}(x)$ with probability at least $1 - \varepsilon$. If an algorithm computes F with bounded error $1/3$, we simply say it computes F with *bounded error*. The choice of $1/3$ here is arbitrary, as the following exercise shows.

Exercise 1.1.2. *Let $\varepsilon, \varepsilon' \in \mathbb{R}$ be such that $0 < \varepsilon' < \varepsilon < 1/2$. Assuming ε is constant, but ε' may vary in n. Fix a quantum algorithm $\mathcal{A}$ that computes a problem F with bounded error ε. Describe and analyze a quantum algorithm that computes F with bounded error ε'. How many calls to $\mathcal{A}$ does your algorithm use (up to constants)?*

Similarly, if $\mathrm{R} \subset \{0,1\}^n \times \{0,1\}^m$ is a relation, we say a quantum algorithm computes R with bounded error ε if for all $x \in \{0,1\}^n$, if Y is the outcome of the algorithm on input x, which is a random variable on $\{0,1\}^m$, $\Pr[(x,Y) \in \mathrm{R}] \geq 1 - \varepsilon$.

One-sided Error Fix $\varepsilon \in (0,1)$. We say a quantum algorithm solves a problem $\mathrm{F} : D \to \{0,1\}$ with *one-sided error* ε if for all $x \in \mathrm{F}^{-1}(0)$,

$$\|\Pi_0 |\psi_T(x)\rangle\|^2 = 1,$$

and for all $x \in \mathrm{F}^{-1}(1)$,

$$\|\Pi_1 |\psi_T(x)\rangle\|^2 \geq 1 - \varepsilon.$$

That is, on input x such that $\mathrm{F}(x) = 0$, the algorithm always outputs 0, and on input x such that $\mathrm{F}(x) = 1$, the algorithm outputs 1 with probability at least $1 - \varepsilon$. If an algorithm computes F with one-sided error $1/3$, we simply say it computes F with *one-sided error*. The choice of $1/3$ here is arbitrary, as in the case of bounded error. We sometimes call bounded error *two-sided* error, to emphasize its distinction from one-sided error.

The decision version of Grover's algorithm, which decides, on input $x \in \{0,1\}^n$ if there exists $i \in [n]$ such that $x_i = 1$, is an example of an algorithm with one-sided error. That's because it only outputs 1 if some $i \in [n]$ is found such that $x_i = 1$. If no such i exists, than none can be found.

Exact and Zero Error A quantum algorithm solves a problem $\mathrm{F} : D \to \{0,1\}^m$ *exactly* if for all $x \in D$,

$$\left\|\Pi_{\mathrm{F}(x)} |\psi_T(x)\rangle\right\|^2 = 1.$$

The Deutsch-Jozsa algorithm and the Bernstein-Vazirani algorithm are examples of exact quantum algorithms.

Exact quantum algorithms should not be confused with *zero-error* quantum algorithms. A zero-error algorithm for F is an algorithm that never outputs an answer that is not $\mathrm{F}(x)$. There are two slightly different ways of looking at this. We could say that the algorithm is allowed to output some $\perp$ symbol indicating it does not know the answer, with probability at most $1/2$, and otherwise, it must output the correct answer. Alternatively, we could consider algorithms that may run forever, but if they terminate, they will output $\mathrm{F}(x)$. Such algorithms don't fit into the model we have described here, since we assume an algorithm always applies T unitaries $U_1, \ldots, U_T$, before outputting. Quantum algorithms, like classical algorithms, can also be allowed to have variable running time. These can be modelled by supposing there is some intermediate measurement applied after each unitary U_t that outputs a bit indicating the algorithm has halted (and so the answer register should be measured) or the algorithm has not halted and the next unitary should be applied. For a formal definition of such algorithms, see [Amb12, Section 3.1] or [Jef22, Section 3].

An example of such an algorithm is Simon's algorithm [dW19, Chapter 3]. Recall that the algorithm works by sampling values $j \in \{0,1\}^n$ until a basis for $\{j : j \cdot s = 0 \mod 2\}$ is found. The expected number of samples needed is $O(n)$, but for any T, there is some finite probability that you have not found a basis yet after T samples (for example, you may have simply measured the same j T times).

A zero-error algorithm's running time is a random variable with no upper bound on its support, so its complexity is measured by the *expected running time*.

Sweeping Errors Under the Rug It is natural to assume quantum algorithms have errors. Even if we make the (incorrect) simplifying assumption that the hardware is perfect, translations between quantum gate sets introduce errors, so for example, if you want to compile the Bernstein-Vazirani algorithm into some specific gate set, the resulting circuit will likely no longer be exact. However, it is often convenient to pretend a quantum algorithm works perfectly. Exercise 1.1.2 helps us justify such an assumption. Roughly speaking, if an algorithm has error probability ε, then any setting in which you call the algorithm $o(1/\varepsilon)$ can't distinguish the algorithm from a perfect one, except with small probability.

1.1.2 Quantum Gates and Circuits

You learned about gates and circuits in [dW19, Section 2.1.2], but we review some basics and notation here. Fix an M-qubit space $H = H_1 \otimes \cdots \otimes H_M$ where for all $i \in [M]$, $H_i \equiv \mathbb{C}^2$. A *local gate* is a unitary on H that acts as the identity on all but at most 2 of the spaces H_i. The choice of 2 here is arbitrary: we could have chosen any constant at least 2. Often these are just called *gates*, but we emphasize "local" to distinguish them from other operations we will sometimes consider "basic operations" that are not local. In practice, it is sufficient to restrict attention to a finite gate set, consisting of a constant number of gate types, applied to all possible choices of qubits. For example, *Clifford+*T is a universal gate set including the following gate types:

$$\mathsf{H} := \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathsf{T} := \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad \mathsf{CNOT} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and sometimes also the redundant gate types:

$$\mathsf{Z} := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \mathsf{T}^4, \quad \mathsf{P} := \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{bmatrix} = \mathsf{T}^2, \quad \mathsf{X} := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \mathsf{HZH}.$$

We can extend these types, which are unitaries acting on 1 or 2 qubits, to act on an M-qubit space, letting $\mathsf{H}(i) := I_2^{\otimes(i-1)} \otimes \mathsf{H} \otimes I_2^{\otimes(M-i)}$ be H applied to the i-th qubit (I_2 is the identity on a single qubit) and similarly for other 1-qubit gates; and $\mathsf{CNOT}(i,j)$ denote CNOT applied so the control is on the i-th qubit, and the target on the j-th qubit.

In theory, we don't usually need to worry about restricting to a particular gate set, since the Solovay-Kitaev theorem allows us to map between different gate sets (see [dW19, Section 2.2]). This mapping does introduce some error, so if we are discussing a setting with no error, than the choice of gate set may actually matter.

In the *quantum circuit model*, quantum algorithms are made up of gates, and the number, T, of gates is the (gate) complexity of the algorithm. In this strict model, the input is often part of the initial state, which does not fit in with our definition of algorithms. However, we can recover this setting by letting the first n gates be conditional X-gates, $\mathsf{X}^{x_i}|0\rangle = |x_i\rangle$, applied to the i-th qubit, so that the first n qubits encode the input.

Quantum Fourier Transforms

We can use local gates to build up more complex computations. One important example of a unitary that can be implemented with a reasonably small number of local gates is a *quantum Fourier transform* – perhaps the *most* important building block of quantum algorithms. A quantum Fourier transform is defined with respect to a *group* G (see Appendix A.1.1). We will restrict our attention to finite Abelian groups. First, consider a cyclic group of the form $\mathbb{Z}_N = \{0, \ldots, N-1\}$ for some positive integer N, which is a group under addition modulo N (all finite Abelian groups are isomophic to direct products of groups of this form). Its quantum Fourier transform $\mathsf{QFT}_{\mathbb{Z}_N}$ acts on $\mathrm{span}\{|x\rangle : x \in \mathbb{Z}_N\}$ as

$$\mathsf{QFT}_{\mathbb{Z}_N} : |x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{z \in \mathbb{Z}_N} \omega_N^{xz} |z\rangle,$$

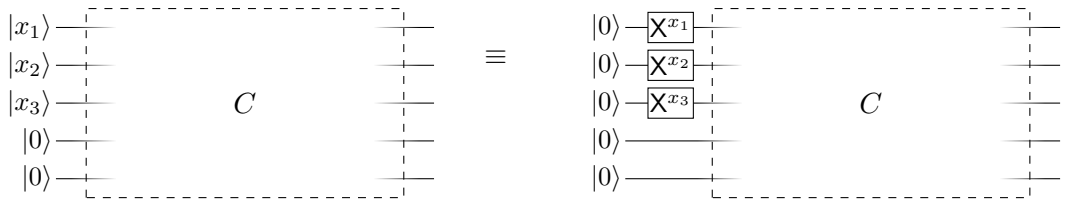


Figure 1.1: A circuit C with the input given on $n = 3$ wires is equivalent to a circuit with all 0s input where the first n gates are input-dependent X^{x_i}s.

where $\omega_N := e^{2\pi i/N}$ is an N-th root of unity. For example, $\mathsf{QFT}_{\mathbb{Z}_2}$ is just a Hadamard gate, H. More generally, we have the following:

Theorem 1.1.1. *For any $N = 2^n$, $\mathsf{QFT}_{\mathbb{Z}_N}$ can be implemented using $O((\log N)^2)$ local gates (see e.g. [dW19, Section 4.5]). Alternatively, for any N, and any constant c, $\mathsf{QFT}_{\mathbb{Z}_N}$ can be approximated with error $1/(\log N)^c$ using $O(\log N \log\log N)$ local gates [HH00].*

We can extend this definition to other finite Abelian groups by defining

$$\mathsf{QFT}_{G_1 \times G_2} = \mathsf{QFT}_{G_1} \otimes \mathsf{QFT}_{G_2},$$

for any pair of finite Abelian groups G_1 and G_2. For example, this gives:

$$\mathsf{QFT}_{\mathbb{Z}_N^\ell}|x\rangle = \frac{1}{\sqrt{N^\ell}} \sum_{z \in \mathbb{Z}_N^\ell} \omega_N^{x \cdot z} |z\rangle,$$

where $x \cdot z = \sum_{i=1}^{\ell} x_i z_i$, and in particular, $\mathsf{QFT}_{\mathbb{Z}_2^\ell} = \mathsf{H}^{\otimes \ell}$.

1.1.3 Quantum Random Access

Classical RAM Quantum circuits can be thought of as the quantum analogue of classical circuits, for example, circuits that use Boolean AND, OR and NOT gates. This gate set is universal for classical computation, since any Boolean function can be expressed as a circuit in these gates. Another universal set of operations is addition and multiplication (over some fixed field). However, when analyzing classical algorithms, we do not only count these basic operations, it is also standard to allow for *random access memory (RAM)* reads and writes. What this means is that if we have a memory of size M (which may depend on the size of the input), we allow an operation that takes any $i \in [M]$, and returns the i-th entry in the memory. This is often assigned unit cost, but there is actually no way to implement this in $O(1)$ gates from any set of gates that each acts on $O(1)$ bits (why?).

A more reasonable cost for this gate would be $O(\log M)$, because we can imagine having a binary tree of depth $\log M$, called a *random access tree*, with a memory cell at each of its M leaves, and a gate at each internal node at depth j that sends an incoming signal encoding index $i \in [M]$ to the left or right child depending on the j-th bit of i, so it eventually reaches the leaf containing x_i, which is triggered to send this bit back up the tree. Note that the total number of gates in this circuit is actually $O(M)$, it's just that only $O(\log M)$ of them were actually activated, so the time required is counted as $O(\log M)$.

In classical computing, RAM gates are important, not only for things like accessing the bits of some large string stored in memory, but also just for very basic computations. For example, you may want to ADD two bits, stored in memory cells i and j. You can do this with a single ADD gate, but only if i and j are fixed in advance of the computation. If i and j are values that are stored in memory, based on computations that have been done so far, then if you want to view the computation as a circuit,

you need to first use RAM read gates to move the contents of cells i and j into some fixed cells, say "1" and "2", and then apply an ADD gate to these cells, and then replace the contents using RAM write gates. In this way, conditional operations that we take for granted as basic steps of a classical computation actually do not fit into the standard circuit model.

Quantum RAM This is similar in quantum computing.

We can define a quantum version of random access read and write gates as follows, where $x \in \{0,1\}^M$, $b \in \{0,1\}$, and $i \in [M]$.

$$\mathsf{READ}|i\rangle|b\rangle|x\rangle = |i\rangle|b \oplus x_i\rangle|x\rangle \tag{1.1}$$

$$\mathsf{WRITE}|i\rangle|b\rangle|x\rangle = |i\rangle|b\rangle|x_1, \ldots, x_{i-1}, x_i \oplus b, x_{i+1}, \ldots, x_M\rangle. \tag{1.2}$$

From this, we can get an important gate called a QRAG:

$$\mathsf{QRAG}|i\rangle|b\rangle|x\rangle = |i\rangle|x_i\rangle|x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_M\rangle. \tag{1.3}$$

Exercise 1.1.3. *Show that* QRAG *can be implemented using 3 calls (total) to* READ *and* WRITE.

As in the case of classical computing, READ, WRITE and QRAG are not local gates – they do not act trivially on all but a constant number of qubits. If we try to argue that the complexity of implementing a QRAG is $O(\log n)$, we must acknowledge that on a single QRAG call, it is possible that all $O(n)$ gates in a random access tree are activated in *some* branch of the superposition (whether this means the cost should be $O(n)$ is a matter for debate). Attempts to give better architectures for quantum random access memory [GLM08] have met with criticism about tolerance to errors [AGJO+15], and the debate about the feasibility continues.

It is not in the scope of this course to decide if implementing such a gate is feasible in practice, but it is important for us to be aware of the controversy surrounding this type of gate. We can do much more if we allow QRAGs than without, so we will often use them, but it is important to be clear when we are working in this model. We must also acknowledge that current quantum computer prototypes cannot implement QRAGs, so computations that require them are not suitable for near-term quantum computers. An exception is when the memory is small, as a QRAG can be implemented in $O(M)$ local gates, but for large memory this is usually unacceptably high.

QCROM A related source of controversy is the feasability of *quantum-accessible classical read-only random access memory*. This is memory that can only store a classical string x, but such that we can query the i-th bit of x, for a superposition of different values $i \in [M]$. That is, we can implement a gate like that in (1.1), with the difference that we assume that the second register contains a classical string x, and not a superposition, so effectively, we assume that if x is stored in QCROM, we can implement the x-dependent map

$$|i\rangle|b\rangle \mapsto |i\rangle|b \oplus x_i\rangle,$$

also on superpositions of different $|i\rangle|b\rangle$. In the past this type of memory has been referred to as QRAM, but we call it QCROM to make the distinction from *quantum* memory with random access gates, and to emphasize that the classical string x is read only. The additional controversy surrounding this model is that in some previous work, a QCROM of size M has been equated with a regular classical memory of size M, along with $\log M$ qubits, and so quantum computations that use just a small quantum memory, in addition to a very large QCROM, have been described as having a small quantum memory and large classical memory, which is not entirely accurate. While there is reason to believe QCROM is easier to implement than fully quantum memory – it need only store a classical state, so, for example, error correction should be easier – it is certainly harder to implement than fully classical memory, as evidenced by the fact that we do not currently know how to make scalable

QCROM. In fact, as far as I am aware, there is currently no implementation of QCROM that requires fewer resources than fully quantum memory.

Again, it is not in the scope of this course to work out the precise difficulty of building QCROM, but it is important for us to be clear about which model our results are in, in order to be precise about resource requirements of our algorithms.

Select Operators An important use of random access memory that we take for granted in classical computation is the ability to read a program stored in memory, and decide what computation to perform. Concretely, we often assume we can look up (or perhaps efficiently compute) what to do at the t-th step of computation – this may be an instruction like $\mathrm{ADD}(i, j)$, meaning add the bit at memory location i into the bit at memory location j.

Analogously, we may use quantum random access gates to implement a *select operator*.

A select operator for unitaries $U_1, \ldots, U_T$ is the unitary

$$U_{\text{select}} = \sum_{t=1}^{T} |t\rangle\langle t| \otimes U_t \tag{1.4}$$

that conditionally applies one of the T different unitaries. **[2]**

Note that this also allows us to apply different unitaries in different branches of a superposition, which we could view as a quantum analogue of doing different classical operations based on some random choice. The complexity of a select operator in the strict quantum circuit model (i.e. in local gates) is generally the sum of the (local) gate complexities of all the U_t. That's because a circuit for U_{select} consists of a sequence of T circuits, for $t = 1$ to T, where the circuit for U_t is implemented in the second register, controlled on the first register containing t. This is certainly not how we generally count costs in classical computation: if we apply one of T classical computations conditioned on the value of some register, the cost we incur is the *maximum* cost of any of the T computations. We can achieve something similar in the quantum case, if we allow random access gates.

Exercise 1.1.4. *Fix unitaries $U_1, \ldots, U_T$ on $\mathbb{C}^{2^M}$ such that for each $t \in [T]$, $U_t = \mathsf{G}_\ell(i, j)$ where G_ℓ is from a constant-sized set of 1- and 2-qubit gate types, $\{\mathsf{G}_\ell\}_{\ell=1}^{c}$, and $i, j \in [M]$ indicate the qubits to which G_ℓ is applied (j is ignored if G_ℓ is a 1-qubit gate). Suppose there is an oracle that indicates the gate type and wires for each t. That is, we can implement a unitary V that acts, for all $t \in [T]$, as:*

$$|t\rangle|0\rangle|0\rangle|0\rangle \mapsto |t\rangle|\ell\rangle|i\rangle|j\rangle$$

where $U_t = \mathsf{G}_\ell(i, j)$. Show how to implement U_{select} as in (1.4) using $O(1)$ calls to V, QRAGs, and local gates.

As a corollary, if we allow QRAGs, then we can efficiently implement U_{select} for gates $U_1, \ldots, U_T$ from a finite gate set, if either of the following is true:

- Descriptions of U_t (gate type and location) are stored in QCROM; or
- descriptions of U_t (gate type and location) are efficiently computable from t. **[3]**

This is similar to conditions needed to be able to run a classical subroutine in a standard classical computer.

1.1.4 Oracles and Quantum Query Complexity

Another basic operation from which we will build up algorithms is an *oracle*, also called a *query operator*. Although oracles may be general unitaries (or even non-unitary operators), we will usually

assume they have one of the following specific forms. Fix a function $f : \{0,1\}^n \to \{0,1\}$. Then there are two natural ways of encoding f in an oracle. We let $\mathcal{O}_f$ be a *standard oracle* for f, which acts on

$$\mathrm{span}\{|x\rangle|b\rangle : x \in \{0,1\}^n, b \in \{0,1\}\}$$

as

$$\mathcal{O}_f : |x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle,$$

and we let $\mathcal{O}_f^{\pm}$ be a *phase oracle* for f, which acts on

$$\mathrm{span}\{|x\rangle : x \in \{0,1\}^n\}$$

as

$$\mathcal{O}_f^{\pm} : |x\rangle \mapsto (-1)^{f(x)}|x\rangle.$$

Note that the phase oracle itself cannot be used to distinguish between a function f and its complement $1 - f$, in which every output bit is flipped, because $\mathcal{O}_f$ and $\mathcal{O}_{1-f} = -\mathcal{O}_f$ are identical up to a global phase. We will assume that we can also implement a *controlled* phase oracle $\mathsf{c}\mathcal{O}_f^{\pm}$ which acts as

$$\mathsf{c}\mathcal{O}_f^{\pm} : |b\rangle|x\rangle = (-1)^{bf(x)}|b\rangle|x\rangle.$$

More generally, if $f : \{0,1\}^n \to \{0,1\}^m$, its standard oracle may be defined by the action $\mathcal{O}_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$, where now $y \in \{0,1\}^m$, and $\oplus$ denotes bitwise xor. For the phase oracle, if we let $\mathcal{O}_f^{\pm}|x\rangle = (-1)^{f(x)}|x\rangle$ (interpreting $f(x)$ as an integer), then it actually only depends on the parity of each $f(x)$, so we are throwing away a large part of f. Instead, a general analog of the controlled phase oracle can be defined defined:

$$\mathsf{c}\mathcal{O}_f^{\pm} : |y\rangle|x\rangle \mapsto (-1)^{y \cdot f(x)}|y\rangle|x\rangle,$$

where $y \in \{0,1\}^m$. Alternatively, we could also define $\mathcal{O}_f|x\rangle|y\rangle = |x\rangle|y + f(x)\rangle$, where we interpret y and $f(x)$ as m-bit integers, and addition is modulo 2^m; and similarly, we could define $\mathsf{c}\mathcal{O}_f^{\pm}|y\rangle|x\rangle = \omega_{2^m}^{yf(x)}$.

In the following exercise, you will show that it doesn't really matter which type of query we consider, standard or phase, as they can be used to implement one another.

Exercise 1.1.5. *Let $G = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_d}$ be a finite Abelian group, and $f : \{0,1\}^n \to [|G|]$. Let $\mathcal{O}_{G,f}$ be defined by the action $\mathcal{O}_{G,f}|x\rangle|y\rangle = |x\rangle|y + f(x)\rangle$, where $y \in G$, we interpret $f(x)$ as an element of G, and $+$ denotes the group operation. Let $\mathsf{c}\mathcal{O}_{G,f}^{\pm}$ be defined by the action $\mathsf{c}\mathcal{O}_{G,f}^{\pm}|x\rangle|y\rangle = \omega_{N_1}^{y_1 f(x)_1} \ldots \omega_{N_d}^{y_d f(x)_d}|x\rangle|y\rangle$, where $y = (y_1, \ldots, y_d) \in G$, and we interpret $f(x) = (f(x)_1, \ldots, f(x)_d)$ as an element of G (for example, if $d = m$ and $N_1 = \cdots = N_d = 2$, $\mathsf{c}\mathcal{O}_{G,f}^{\pm}|x\rangle|y\rangle = (-1)^{y \cdot f(x)}|x\rangle|y\rangle$).*

1. *Show that $\mathcal{O}_{G,f}$ can be implemented using one call to $\mathsf{c}\mathcal{O}_{G,f}^{\pm}$ and one application each of QFT_G and its inverse.*
2. *Show that $\mathsf{c}\mathcal{O}_{G,f}^{\pm}$ can be implemented using one call to $\mathcal{O}_{G,f}$ and one application each of QFT_G and its inverse.*

Oracles as Input We often use an oracle as a way of abstracting the input to a problem. An important example is *oracular search*, which is the problem solved by Grover's algorithm, defined as follows:

Problem: SEARCH$_{2^n}$

Input: An oracle $\mathcal{O}_f$ for $f : \{0,1\}^n \to \{0,1\}$.
Output: 1 if there exists $i \in \{0,1\}^n$ such that $f(i) = 1$, and 0 else.

Here f is the input (given by an oracle), not the problem to be solved, which is sometimes confusing. We could equivalently consider the input to be a string $x \in \{0,1\}^N$ where $N = 2^n$ (so $[N] \equiv \{0,1\}^n$), and for each $i \in [N]$, x_i takes the place of $f(i)$. Either way, by abstracting the input as we have done, this problem can model multiple settings, including:

1. The input is a string x that is actually stored in memory somewhere, in which case, $\mathcal{O}_f$ can be implemented using a random access read, READ as in (1.1).

2. The input is some function f that is described in some way (we have abstracted away the details) that makes it easy to compute $f(i)$ for any i. For example, if f is described by a formula on n bits, then oracular search decides if there is a satisfying assignment. Or if there is some string $z \in [q]^N$ stored in random access memory, we could define $f : \{0,1\}^{2\log N} \to \{0,1\}$ so that $f(i,j) = 1$ if and only if $z_i = z_j$, and then oracular search on input f decides a problem called *element distinctness*.

Another example is when the input to a problem is a graph, and we want to decide some property of the graph. There are multiple possible ways we could be given a graph as input. We could, for example, assume we have an oracle $\mathcal{O}_G$ that acts as $\mathcal{O}_G|u,i\rangle|0\rangle \mapsto |u,i\rangle|f_u(i)\rangle$, where $f_u(i) \in V$ is the i-th neighbour of the vertex u in G. This implicitly assumes we have some way of efficiently computing this, either because of some nice structure in G, or because G is simply given as an array of n arrays of neighbours.

Quantum Query Complexity For any problem, $\mathrm{F} : \{0,1\}^N \to \{0,1\}^m$, we can view its input as an oracle $\mathcal{O}_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle$ for some $x \in \{0,1\}^N$, and consider algorithms for F whose unitaries U_t are restricted to being either: (1) an oracle query $U_t = \mathcal{O}_x$; or (2) an input-independent unitary – we call this a *quantum query algorithm*. If we don't impose any restrictions on the input-independent unitaries, then without loss of generality, we may assume that the oracle queries are exactly those U_t for which t is even. Then the number of oracle queries made by the algorithm is $T/2$, and we call this the *query complexity* of the algorithm.

We can define the *bounded-error quantum query complexity of* F, $\mathsf{Q}(\mathrm{F})$ as the minimum query complexity of any quantum query algorithm that computes F with bounded error. We can define similar quantities for one-sided error, or exact algorithms, but the bounded-error case is the most standard.

Quantum query complexity is studied in part because it is much easier to only count queries than to count every operation, but results in query complexity can also be meaningful. For example, a lower bound on $\mathsf{Q}(\mathrm{F})$ is also a lower bound on the total number of operations (time complexity) required to compute F. While the time complexity – in which we count *all* basic operations, not just queries – and query complexity can differ by a lot (the quantum query complexity can never be more than the input size), for certain problems they are very close. Moreover, while upper bounds on quantum query complexity are not meaningful in practice, since they say nothing about the time complexity, it has often been the case that query upper bounds have been the *first step* to showing more practically meaningful upper bounds on time complexity.

Oracles Abstracting Subroutines Oracles are different from other gates, in that, not only are they not generally local, we don't always assume they can be implemented as a "basic" operation. Rather, we sometimes use an oracle $\mathcal{O}_f$ to abstract away the details of some *subroutine* computing f. It may be that the cost of this subroutine is not negligible, but we want to consider it as a single operation so that we don't have to think about its details. In that case, to obtain the true complexity, we need to account for this cost. Naively, this means counting every application of $\mathcal{O}_f$ as T, if T is the complexity of the subroutine, however, we can sometimes do better. For example, if $\mathcal{O}_f$ is applied to a state $\sqrt{1-\alpha^2}|0\rangle|\psi_0\rangle + \alpha|1\rangle|\psi_1\rangle$ controlled on the first qubit, then this need only cost $|\alpha|^2 T$ – a fact that is not at all obvious. A corollary is that if we apply $\sum_i |i\rangle\langle i| \otimes \mathcal{O}^{(i)}$ to a state $\sum_i \alpha_i|i\rangle|\psi_i\rangle$, where T_i is the complexity of $\mathcal{O}^{(i)}$, the cost is the *average* $\sum_i |\alpha_i|^2 T_i$ (see [Jef22, BJY23]).

2.1 Hidden Subgroup Problems

In this section, we will study one of the earliest and most important quantum algorithms, which solves *the hidden subgroup problem* – actually a type of problem, parametrized by a family of groups – for *Abelian groups*. This section requires some basic group theory, which you can also find in Appendix A.1.1. Parts of this section borrow heavily from [dW19, Chapter 6].

2.1.1 Period Finding

Before looking at the general case, let us review an algorithm for a problem you have probably seen before: period finding. In fact, you may recall from [dW19, Chapter 5] that factoring can be reduced to period finding, which is solved by the main quantum subroutine in Shor's algorithm.

The *period* of a function f on the additive group $\mathbb{Z}_N$ (the integers modulo N) is the smallest positive integer r such that $f(x) = f(x+r)$ for all $x \in \mathbb{Z}_N$ (in other words, f repeats every r steps. Note that this is always true for $r = N$, since x and $x + N$ are the same element of $\mathbb{Z}_N$, r can also be smaller than N.

Exercise 4.1. *What are the possible values of the period r of a function on $\mathbb{Z}_N$?*

We will call a function f on $\mathbb{Z}_N$ *one-to-one-periodic* if it is one-to-one on the set $\{0, \ldots, r-1\}$, where r is the period (that is, it takes r distinct values, and then repeats the sequence. For such a function, we have:

$$f(x) = f(x') \Leftrightarrow \exists h \in \mathbb{Z} \text{ s.t. } x - x' = hr.$$

Problem: $\textsc{Period}_N$

Input: An oracle $\mathcal{O}_f$ for a one-to-one-periodic function $f : \mathbb{Z}_N \to S$ for some finite set S.
Output: The period r of f.

The group $\mathbb{Z}_N$ is a cyclic group, meaning it is generated by a a single element. Any cyclic group of size N is isomorphic to $\mathbb{Z}_N$. By Lagrange's theorem (see Appendix A.1.1), any subgroup H of $\mathbb{Z}_N$ must be of a size that divides N. In particular, we have for any $M \leq N$ such that $M|N$, the set

$$\{0, M, 2M, \ldots, N - M\}$$

is a subgroup of $\mathbb{Z}_N$ of size N/M, and all subgroups have this form.

Algorithm 1. Period Finding

1. Initialize two registers of dimension N and $|S|$ to $|0\rangle|0\rangle$.
2. Apply $\mathsf{QFT}_{\mathbb{Z}_N}$ to the first register to get $\frac{1}{\sqrt{N}} \sum_{z=0}^{N-1} |z\rangle|0\rangle$.
3. Apply $\mathcal{O}_f$ to get $\frac{1}{\sqrt{N}} \sum_{z=0}^{N-1} |z\rangle|f(z)\rangle$.
4. Measure the second register to get some value $y \in S$, and the remaining state $\frac{1}{\sqrt{N/r}} \sum_{h=0}^{N/r-1} |x + hr\rangle$ for some x.
5. Apply $\mathsf{QFT}_{\mathbb{Z}_N}$, and then measure and output the result.

Lemma 2.1.1. *Algorithm 1 outputs a uniformly random element of $\{0, N/r, 2N/r, \ldots, (r-1)N/r\}$.*

Proof. Upon measuring $y \in S$ in step 4, the remaining state is proportional to:

$$\sum_{z \in \mathbb{Z}_N : f(z)=y} |z\rangle.$$

Let $x \in \mathbb{Z}_N$ be such that $f(x) = y$ (we know such an x exists, or we would not have measured y). Then the $z : f(z) = y$ are precisely those z such that $x - z = hr$ for some integer h. Thus, the state remaining is (now we write it with normalization):

$$\frac{1}{\sqrt{N/r}} \sum_{q=0}^{N/r-1} |x + hr\rangle,$$

as claimed. Applying $\mathsf{QFT}_{\mathbb{Z}_N}$, we get:

$$\begin{aligned}
\mathsf{QFT}_{\mathbb{Z}_N} \frac{1}{\sqrt{N/r}} \sum_{h=0}^{N/r-1} |x + hr\rangle &= \sqrt{\frac{r}{N}} \sum_{h=0}^{N/r-1} \frac{1}{\sqrt{N}} \sum_{z=0}^{N-1} \omega_N^{z(x+hr)} |z\rangle \\
&= \sqrt{\frac{r}{N}} \frac{1}{\sqrt{N}} \sum_{h=0}^{N/r-1} \sum_{z=0}^{N-1} \omega_N^{zx} \omega_N^{zhr} |z\rangle \\
&= \sqrt{\frac{r}{N}} \frac{1}{\sqrt{N}} \sum_{z=0}^{N-1} \omega_N^{zx} \left(\sum_{h=0}^{N/r-1} \omega_N^{zhr} \right) |z\rangle.
\end{aligned}$$

Suppose $z \in \mathbb{Z}_N$ is such that $\frac{N}{r} | z$. Then $N | zr$, so $N | zrh$ for all h, and so

$$\omega_N^{zhr} = e^{2\pi i z h r / N} = (e^{2\pi i})^{zrh/N} = 1,$$

which implies

$$\sum_{h=0}^{N/r-1} \omega_N^{zhr} = N/r.$$

On the other hand, suppose $\frac{N}{r}$ does not divide z. Then you will show in Exercise 4.2 that

$$\sum_{h=0}^{N/r-1} \omega_N^{zhr} = \sum_{h=0}^{N/r-1} \omega_{N/r}^{zh} = 0.$$

Thus

$$\mathsf{QFT}_{\mathbb{Z}_N} \frac{1}{\sqrt{N/r}} \sum_{h=0}^{N/r-1} |x + hr\rangle = \sqrt{\frac{r}{N}} \frac{1}{\sqrt{N}} \sum_{z'=0}^{r-1} \omega_N^{z'(N/r)x} (N/r) |z'(N/r)\rangle = \frac{1}{\sqrt{r}} \sum_{z'=0}^{r-1} \omega_r^{z'x} |z'(N/r)\rangle.$$

Since $|\omega_r^{z'x}| = 1$, these do not impact the measurement statistics, so measuring this state yields a uniformly random element of $\{0, N/r, 2N/r, \ldots, (r-1)N/r\}$. □

Exercise 4.2. *Let M and z be positive integers such that M does not divide z. Show that*

$$\sum_{h=0}^{M-1} \omega_M^{zh} = 0.$$

Hint: You can write $z = Mq + \ell$ for some integers q and ℓ such that $\ell \in \{1, \ldots, M-1\}$.

Running a single iteration of Algorithm 1 doesn't fully solve the period finding problem. If you obtain a single value $kN/r = (2k)N/(2r)$, you cannot distinguish between period r, and period $2r$. However, given a small number of samples, we can recover a value that is the correct period with high probability. We discuss the high-level idea. Suppose you first measure kN/r, from which you can recover $a_1 = r/k$. Note that $a_1|N$ since $r|N$. Then you have narrowed down the value of r to all the multiples of a_1:

$$r \in \{a_1, 2a_1, \ldots, N - a_1\} = K_1,$$

which is a subgroup of $\mathbb{Z}_N$. Suppose we measure a second value $k'N/r$ to get a second $a_2 = r/k'$. Then we have narrowed down

$$r \in \{a_1, 2a_1, \ldots, N - a_1\} \cap \{a_2, 2a_2, \ldots, N - a_2\} = K_1 \cap K_2.$$

The intersection of two subgroups is also a subgroup, and in this case,

$$K_1 \cap K_2 = \{\ell, 2\ell, \ldots, N - \ell\}$$

where ℓ is the least common multiple of a_1 and a_2. If $K_1 \subseteq K_2$, which happens precisely when a_1 is a multiple of a_2, then $K_1 \cap K_2 = K_1$, and so a_2 was not helpful at all. Otherwise, $K_1 \cap K_2$ is a proper subgroup of K_1, meaning $|K_1 \cap K_2| < |K_1|$. By Lagrange's theorem, $|K_1 \cap K_2|$ must divide $|K_1|$, and so $|K_1 \cap K_2| \leq \frac{1}{2}|K_1|$. Thus, we have divided the set of possibilities for r in half. After a logarithmic number of such moves, we will end up with

$$K_1 \cap K_2 \cap \cdots = \{r, 2r, \ldots, N - r\},$$

which will never get any smaller, and from which we can recover r. We make this more precise for a more general case in the proof of Lemma 2.1.8.

2.1.2 Hidden Subgroup Problems

We have just seen how quantum algorithms can find periodic structure over cyclic groups. A one-to-one-periodic function over $\mathbb{Z}_N$ with period r identifies the subgroup $\langle r \rangle = \{0, r, 2r, \ldots, N - r\}$ with a single value, $f(0)$ – we can think of this as a *colour* – and for each shift of the subgroup $\langle r \rangle$,

$$a + \langle r \rangle := \{a, r + a, 2r + a, \ldots, N - r + a\},$$

f assigns a distinct colour.

Not all groups are cyclic. However, for *any* subgroup H of *any* group G, H induces a generalization of this periodic structure, with shifts of H (called cosets) forming the entire group. For any subgroup H of G, and $g \in G$, the coset $g + H$ is the set:

$$g + H := \{g + h : h \in H\}.$$

It is not difficult to see that for all $g \in G$, $|g + H| = |H|$.

Exercise 4.3. *Let H be a subgroup of G, and $g, g' \in G$. Show that if $g - g' \in H$, $g + H = g' + H$, and otherwise, $g + H$ and $g' + H$ are distinct.*

What Exercise 4.3 says is that the distinct cosets of H form a partition of G into $|G|/|H|$ disjoint sets of size $|H|$. This is illustrated in Figure 2.2. Hidden subgroup problems generalize period finding from cyclic groups to this more general structure.

Hidden subgroup problems are parametrized by a family of groups. For a family of groups G, the associated hidden subgroup problem is defined:

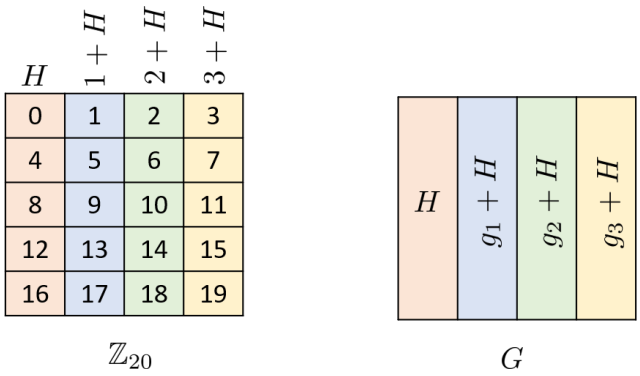


Figure 2.2: On the left, the group $\mathbb{Z}_{20}$, and the subgroup $H = \langle 4 \rangle$, shown with all its shifts. On the right, a general group and subgroup induces a similar partition of G into the shifts of H.

Problem: HSP_G

Input: An oracle $\mathcal{O}_f$ for $f : G \to S$ for some finite set S, with the promise that there is a subgroup H of G such that $f(x) = f(y)$ if and only if $x + H = y + H$. That is, f takes a distinct value on each coset of H.

Output: A set of generators for $H^\perp$.

At first glance, it would have been more natural to output a set of generators for H. However, given a set of generators for $H^\perp$ (which is what our algorithm will naturally find), it is possible to efficiently compute a set of generators for H, using standard linear algebra techniques; we will see this in specific applications.

Before we discuss quantum algorithms for this problem, an important question is how hard the problem is for classical computers. This depends on G, but we have the following (see [Chi21, Theorem 5.1]):

Theorem 2.1.2. *Suppose G has a set of K subgroups whose only common element is the identity. Then the randomized query complexity of HSP_G is $\Omega(\sqrt{K})$.*

Note that there are groups G for which HSP_G is easy for a classical computer. For example, when p is prime $\mathbb{Z}_p$ only has two subgroups, $\{0\}$ and $\mathbb{Z}_p$ itself, and a classical algorithm can easily distinguish these two cases. Some instances of note that are easy for a quantum computer and believed to be hard classically are mentioned in Section 2.1.5, Section 2.1.6 and Section 2.1.7.

Exercise 2.1.1. *Describe a classical algorithm for HSP_G when $G = \mathbb{Z}_{2^n}$ for some $n \in \mathbb{N}$ that uses only poly(n) queries to $\mathcal{O}_f$.*

Exercise 2.1.2. *What are the possible sizes of S in the definition of HSP_G?*

We will soon see an efficient quantum algorithm for HSP_G whenever G is Abelian. First, we need some concepts from representation theory, which studies symmetries (patterns) in groups.

2.1.3 Representations

A representation of a group G (see Appendix A.1.1) is a map $\rho : G \to \mathcal{GL}_n(\mathbb{C})$, where $\mathcal{GL}_n(\mathbb{C})$ is the set of invertible linear transformations of $\mathbb{C}^n$ (also called the *general linear group*), such that for all $g, g' \in G$, $\rho(g + g') = \rho(g)\rho(g')$ **[4]**

(i.e., ρ is a *group homomorphism*). If $n = 1$ (so $\mathcal{GL}_n(\mathbb{C}) = \mathbb{C}$), then ρ is called a *character*.

Character: representation $\chi : G \to \mathbb{C}$

We usually use the letter χ for characters. For any character χ of G and $g \in G$, we must have $|\chi(g)| = 1$, since $|\chi(g^k)| = |\chi(g)|^k$ for all $k \in \mathbb{Z}$. Also, since a character χ of G is also a group homorphism when restricted to any subgroup H of G, its restriction to H is a character of H.

Example 2.1.3. For any G, the map $\chi : G \to \mathbb{C}$ defined $\chi(g) = 1$ for all $g \in G$ is a character.

Example 2.1.4. Let $G = \mathbb{Z}_N$ for an even number N, and define $\chi(g) = (-1)^g$. Then χ is a character.

Example 2.1.5. Let $G = \mathbb{Z}_N$ for any $N \in \mathbb{N}$, and define $\chi(g) = \omega_N^g = e^{2i\pi g/N}$. Then χ is a character.

We let $\widehat{G}$ denote the set of all characters of G, which is a group under pointwise multiplication: $(\chi \cdot \chi')(g) = \chi(g)\chi'(g)$. We call $\widehat{G}$ the *dual group*.

Dual group: $\widehat{G} = \{\chi : \chi \text{ is a character of } G\}$,
a group under pointwise multiplication $(\chi \cdot \chi')(g) = \chi(g)\chi'(g)$

Suppose $G = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_\ell}$ is a finite Abelian group, and as usual, let $\omega_{N_j} = e^{2i\pi/N_j}$ be an N_j-th root of unity. For any $g = (g_1, \ldots, g_\ell), x = (x_1, \ldots, x_\ell) \in G$, let

$$\chi_g(x) = \omega_{N_1}^{g_1 x_1} \ldots \omega_{N_\ell}^{g_\ell x_\ell}, \tag{2.5}$$

from which it is immediate that $\chi_g(x) = \chi_x(g)$. Then it is easy to verify that χ_g is a character.

Exercise 2.1.3. *Show that for any $g \in G = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_\ell}$, χ_g as defined in (2.5) is a character of G.*

Moreover, it turns out all characters have this form:

$$\widehat{G} = \{\chi_g : g \in G\},$$

and for all $g, g', x \in G$, $\chi_g(x)\chi_{g'}(x) = \chi_{g+g'}(x)$, so the map $g \mapsto \chi_g$ is a group isomorphism from G to $\widehat{G}$. Referring to Section 1.1.2, we can see that

for any finite Abelian group G, we have $\mathsf{QFT}_G|g\rangle = \frac{1}{\sqrt{|G|}} \sum_{h \in G} \chi_g(h)|h\rangle =: |\chi_g\rangle$

It is easy to verify that the characters $\{|\chi_g\rangle\}_{g \in G}$ form an orthonormal basis, which is necessary and sufficient for QFT_G to be unitary.

For a character $\chi \in \widehat{G}$, define its kernel:

$$\ker \chi = \{g \in G : \chi(g) = 1\}$$

Finally, for a subgroup H of G, define

$$H^\perp = \{\chi_g : \chi_g(h) = 1 \text{ for all } h \in H\}$$

We can show a correspondance between the elements of $H^\perp$ and the cosets of H, to prove the following.

Lemma 2.1.6. $|H^\perp| = \frac{|G|}{|H|}$.

Proof. Let ϕ be the map from $H^\perp$ to cosets of H, defined by mapping χ_g to the coset $g+H$. We will show that this map is a bijection, establishing the claim in the lemma statement, since the cosets of H are a partition of G into $|G|/|H|$ disjoint sets.

Let $\chi_g, \chi_{g'} \in H^\perp$ for distinct g and g'. Then for all $h \in H$,

$$1 = \chi_h(g)\chi_h(g')^{-1} = \chi_h(g-g').$$

If this is true for $g-g' \in H$, then this would imply $g-g'$ is the identity, but that is a contradiction, since g and g' are distinct. Thus $g-g' \notin H$, meaning $g+H$ and $g'+H$ are different cosets, establishing that ϕ is injective. We leave it as an exercise to establish that every coset $x+H$ contains some g such that $\chi_g \in H^\perp$, proving that ϕ is surjective. □

2.1.4 Quantum Algorithm for Abelian Hidden Subgroup

When G is Abelian, there is an efficient quantum algorithm for HSP_G. The following algorithm was first described by [Kit96] and worked out further by [ME98].

Algorithm 2. Dual character

1. Initialize two registers of dimensions $|G|$ and $|S|$ to $|0\rangle|0\rangle$.
2. Apply QFT_G to the first register to get $\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|0\rangle$.
3. Apply $\mathcal{O}_f$ to get $\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|f(g)\rangle$.
4. Measure the second register to get some value $y \in S$, and the remaining state $|x+H\rangle := \frac{1}{\sqrt{|H|}}\sum_{h\in H}|x+h\rangle$ where x is any value in $f^{-1}(y)$.
5. Apply QFT_G, and then measure, to get some g, and output χ_g.

We can implement QFT_G in $O(\log^2|G|)$ local gates, so the total complexity of Algorithm 2 is $O(\log^2|G|)$ local gates, plus a single query to $\mathcal{O}_f$.

Lemma 2.1.7. *Algorithm 2 outputs a uniformly random element of $H^\perp$.*

Proof. Upon measuring $y \in S$ in step 4, the remaining state is proportional to:

$$\sum_{g\in G: f(g)=y} |g\rangle.$$

By assumption on f, $\{g \in G : f(g) = y\}$ is a coset of H, so for any $x \in G : f(x) = y$, that coset can be written $x+H$, so we are left with the state $|x+H\rangle$, as claimed. Applying QFT_G, we get:

$$\begin{aligned}
\mathsf{QFT}_G|x+H\rangle &= \frac{1}{\sqrt{|H|}}\sum_{h\in H}\frac{1}{\sqrt{|G|}}\sum_{z\in G}\chi_{x+h}(z)|z\rangle \\
&= \frac{1}{\sqrt{|H||G|}}\sum_{h\in H}\sum_{z\in G}\chi_x(z)\chi_h(z)|z\rangle \\
&= \frac{1}{\sqrt{|H||G|}}\sum_{z\in G}\chi_x(z)\left(\sum_{h\in H}\chi_h(z)\right)|z\rangle.
\end{aligned}$$

Suppose $\chi_z \in H^\perp$. Then $\chi_z(h) = 1$ for all $h \in H$, so $\chi_h(z) = 1$ for all $h \in H$, and so:

$$\sum_{h\in H}\chi_h(z) = |H|.$$

On the other hand, suppose $\chi_z \in \widehat{G} \setminus H^\perp$. We have that χ_z restricted to H is a character of H, and since it is not in $H^\perp$, its restriction to H is not the character that takes value 1 on all of H, and so it must be orthogonal to this character, meaning:

$$\sum_{h \in H} \chi_h(z) = \sum_{h \in H} \chi_z(h) = 0.$$

Thus:

$$\mathsf{QFT}_G|x + H\rangle = \frac{1}{\sqrt{|H||G|}} \sum_{z \in G : \chi_z \in H^\perp} \chi_x(z)|H||z\rangle = \sqrt{\frac{|H|}{|G|}} \sum_{z \in G : \chi_z \in H^\perp} \chi_x(z)|z\rangle.$$

Since $|\chi_x(z)| = 1$ for all x and z, measuring this state yields a uniformly random $z \in G$ such that $\chi_z \in H^\perp$. □

When we find some $\chi_g \in H^\perp$, we have narrowed down the possibilities for H, because we know that H must be a subset of

$$\ker(\chi_g) = \{g' \in G : \chi_g(g') = 1\}.$$

If we find sufficiently many elements of $H^\perp$, then H is just the intersection of their kernels. This is the idea behind the following algorithm.

Algorithm 3. Abelian HSP

1. $L \leftarrow \emptyset$.
2. Repeat $4 \log |G|$ times:
   (a) Call Algorithm 2 to obtain χ, and add it to L.
3. Output L.

Since Algorithm 2 uses $O(\log^2 |G|)$ local gates and queries, Algorithm 3 uses $O(\log^3 |G|)$ local gates and queries.

Lemma 2.1.8. *Let $K_L = \bigcap_{\chi \in L} \ker(\chi)$. After $4 \log |G|$ repetitions in Algorithm 3, $K_L = H$ with probability at least $1 - |G|^{-\frac{1}{2\ln(2)}} \approx 1 - |G|^{-0.72}$.*

Proof. Suppose $K_L \neq H$. Then H is a proper subgroup of K_L, so $|H| < |K_L|$. By Lagrange's theorem (see Appendix A.1.1), $|H|$ divides $|K_L|$, so $|H| \leq |K_L|/2$. Since $|H^\perp| = |G|/|H|$, the probability that the next uniformly sampled $\chi \in H^\perp$ is such that $K_L \subseteq \ker \chi$ – i.e., is in $K_L^\perp$ – is:

$$\frac{|K_L^\perp|}{|H^\perp|} = \frac{|H|}{|G|}|K_L^\perp| = \frac{|H|}{|G|}\frac{|G|}{|K_L|} \leq \frac{1}{2}.$$

Thus, with probability at least $1/2$, we sample χ such that $|K_L \cap \ker(\chi)| < |K_L|$, and then, again by Lagrange's theorem, $|K_L \cap \ker(\chi)| \leq |K_L|/2$. Thus, each time we add an element to L, with probability at least $1/2$, we reduce the size of K_L by at least $1/2$. The number of times this occurs is a binomial random variable (see Appendix A.1.2) $B(k, p)$ with $k = 4 \log |G|$ samples, and success probability $p \geq 1/2$, so by Hoeffding's inequality (see (A.12)), the probability that there are at least $z = \log(|G|/|H|)$ successes is at least:

$$\begin{aligned} 1 - e^{-2k(p-z/k)^2} &\geq 1 - e^{-2k\left(\frac{1}{2} - \frac{\log|G| - \log|H|}{4\log|G|}\right)^2} \geq 1 - e^{-2k\left(\frac{1}{2} - \frac{\log|G|}{4\log|G|}\right)^2} \\ &\geq 1 - e^{-2k\left(\frac{1}{4}\right)^2} = 1 - e^{-k/8} = 1 - e^{-\frac{1}{2}\log|G|} = 1 - e^{-\frac{1}{2}\frac{\ln|G|}{\ln 2}} = 1 - |G|^{-\frac{1}{2\ln(2)}}. \end{aligned}$$

□

2.1.5 Application to Factoring

Kitaev's algorithm for Abelian HSP was inspired by Shor's factoring algorithm, which follows as a special case. Recall from [dW19, Chapter 5] how factoring can be reduced to period finding, which we also saw in Section 2.1.1, and which is a special case of HSP. If M (called N in [dW19] is the number we want to factor, then $\mathbb{Z}_M^*$ is a cyclic group of order $N = \phi(M)$ (see Appendix A.1.1), so it's isomorphic to $\mathbb{Z}_N$. If $\gcd(x, M) = 1$, then in order to factor, we want to find the period of the function $f(a) = x^a \mod M$. Since f is efficiently computable, by repeated squaring, we can efficiently implement $\mathcal{O}_f$.

Note that we have no guarantee that $x^a \mod M$ is one-to-one on its period. Nonetheless, using the method of continued fractions, it is still possible to recover the period using samples from Algorithm 1. See [dW19, Section 5.4] for details.

Exercise 2.1.4. *Let G and f be as above, and let $H = \langle r \rangle$. Show that for any $a, b \in \mathbb{Z}_{\phi(M)}$, $f(a) = f(b)$ if and only if $a - b \in H$. Thus f hides H.*

We have

$$\begin{aligned} H^\perp &= \{\chi_g : g \in \mathbb{Z}_{\phi(M)}, \chi_g(h) = 1,\ \forall h \in \langle r \rangle\} \\ &= \{\chi_g : g \in \mathbb{Z}_{\phi(M)}, e^{2\pi i g h/\phi(M)} = 1,\ \forall h \in \langle r \rangle\} \\ &= \{\chi_g : g \in \mathbb{Z}_{\phi(M)}, e^{2\pi i g c r/\phi(M)} = 1,\ \forall c \in \mathbb{Z}\} \\ &= \{\chi_g : g \in \mathbb{Z}_{\phi(M)}, gr = 0 \mod \phi(M)\}, \end{aligned}$$

which is the set of integer multiples of $\phi(M)/r$. By taking the GCD of a small number of samples from this set, we can recover $\phi(M)/r$, from which we can recover r, assuming $\phi(M)$ is known.

Recall that in the setting of factoring, we are given M, but we do not know $\phi(M)$ (in many cases, computing $\phi(M)$ is as hard as factoring M). This means that we actually cannot run Algorithm 2 as stated, because we can't implement $\mathsf{QFT}_{\mathbb{Z}_{\phi(M)}}$ without knowing $\phi(M)$. The solution is to instead use $\mathsf{QFT}_{\mathbb{Z}_q}$ for $q \approx M^2$. This is precisely what you have seen in Shor's algorithm ([dW19, Chapter 5]), and the following exercise asks you to put this in the language of HSP.

Exercise 2.1.5. *Let $q = 2^{2 \log M}$. Prove that by running Algorithm 2 a small number of times, using $\mathsf{QFT}_{\mathbb{Z}_q}$ instead of $\mathsf{QFT}_{\mathbb{Z}_{\phi(M)}}$ (with $f(a) = x^a \mod M$), you can recover the period r of x with probability at least $2/3$.*

2.1.6 Application to Discrete Log

Another number theoretic problem whose hardness is the basis of important and widely used cryptosystems, such as Diffie-Hellman [DH76], is the discrete log problem. For a family of cyclic multiplicative[1] groups C, the associated discrete log problem is

Problem: $\textsc{Dlog}_C$

Input: $g, A \in C$ such that $\langle g \rangle = C$
Output: $a \in \mathbb{Z}$ such that $g^a = A$.

The condition $\langle g \rangle = C$ means that g is a generator of C. A cyclic group is precisely a group that can be generated by a single element, so such a g must exist. It is generally believed that no classical algorithm can solve this problem in $\text{poly}(\log N)$ time, where $N = |C|$, with the best known classical algorithm requiring $2^{\Theta(\sqrt{\log N})}$ time.

[1] This just means we write the group operation as a multiplication: gh instead of $g + h$.

Shor's original paper on quantum factoring also described how to solve discrete log (this application apparently came first), and it also follows as a case of Abelian HSP, as follows.

Let $G = \mathbb{Z}_N \times \mathbb{Z}_N$, and $f : G \to C$ be defined $f(x, y) = g^x A^{-y} = g^x g^{-ya}$.

Note that f is efficiently computable by repeated squaring, meaning we can efficiently implement $\mathcal{O}_f$. Then we have:

$$\begin{aligned} & f(x_1, y_1) = f(x_2, y_2) \\ \Leftrightarrow \quad & g^{x_1 - y_1 a} = g^{x_2 - y_2 a} \\ \Leftrightarrow \quad & x_1 - x_2 = (y_1 - y_2)a \mod N \qquad \text{since } C = \langle g \rangle = \{g^0, g^1, \ldots, g^{N-1}\}. \end{aligned}$$

This is satisfied precisely when $y_1 - y_2 = q$ for some $q \in \mathbb{Z}_N$, and $x_1 - x_2 = qa$; that is,

$$(x_1, y_1) - (x_2, y_2) = (qa, q).$$

Thus f hides the subgroup $H = \langle (a, 1) \rangle$

and so finding the unique generator is equivalent to finding a. More than that, we have the following:

Exercise 2.1.6. *Let $H = \langle (a, 1) \rangle \subset \mathbb{Z}_N \times \mathbb{Z}_N$, as above. Show that $H^\perp = \{\chi_{(c,-ac)} : c \in \mathbb{Z}_N\}$.*

Thus, even just sampling one element of $H^\perp$, as is done in Algorithm 2, is sufficient to solve $\textsc{Dlog}(g, A)$, as long as c is invertible in $\mathbb{Z}_N$, which happens precisely when c and N are coprime. This happens with probability at least $\phi(N)/N = \Omega\left(\frac{1}{\log \log N}\right)$, where ϕ is Euler's totient.

2.1.7 Further Directions

Algorithm 3 for HSP_G over finite Abelian groups can also be extended to infinite Abelian groups, such as $\mathbb{R}$, where it can be used to give a quantum algorithm for *Pell's equation* and the *Principal ideal problem* [Hal07, BS16].

There is no known efficient quantum algorithm for HSP_G over non-Abelian groups, aside from some special cases, including when H is a *normal* subgroup of G [HRTS03]; when G is *solvable* [Wat01, IMS03]; when G is *nil-2* [ISS12].

An efficient algorithm for general non-Abelian HSP would have some very interesting consequences. When $G = S_n$, the symmetric group, then the problem of *graph isomorphism* can be reduced to HSP_G (see [dW19, Section 6.3.1]). Until somewhat recently, this was believed to require $2^{\text{poly}(n)}$ time classically, but in 2015 Babai gave a *quasi-polynomial* $2^{\text{poly}(\log n)}$ time classical algorithm for graph isomorphism [Bab16].

Since factoring and discrete log are easy for quantum computers, new cryptosystems based on the hardness of other problems need to be developed and standardized. This process is well underway, and the front runner for a hard problem – believed to be hard even for quantum computers – is a problem called the *shortest vector problem*, which asks to find the shortest vector in a lattice – the set of integer linear combinations of some real vectors – given a random basis. A version of this problem (the unique shortest vector problem) could be efficiently solved if an algorithm like our Algorithm 2 (called the "standard method") could solve HSP_G over the *dihedral group*, D_n, which is the group of symmetries of an n-sided regular polygon [Reg04a]. More details can be found in [Chi21, Chapter 10]. While no polynomial-time quantum algorithm is known for HSP_{D_n}, Kuperberg [Kup05] gave an algorithm that runs in sub-exponential time $2^{O(\sqrt{n})}$. Later Regev [Reg04b] gave a sub-exponential time algorithm that uses only polynomial space.

While no efficient algorithm is known for HSP_G for non-Abelian groups in general, there is a *query-efficient* algorithm, which uses $\text{poly}(\log|G|)$ queries, but $\text{poly}(|G|)$ total operations [EHK04] (or see [dW19, Section 6.3.3]). An implication of this is that while we believe HSP_G is hard for some non-Abelian groups, we will not be able to prove that using a query lower bound (which is bad news, since that is our main way of proving things are hard for quantum computers).

A good source for more details on these directions is [Chi21].

2.2 Phase Estimation

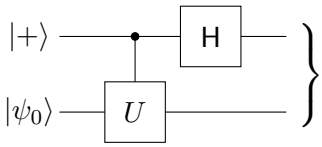
Suppose someone gives you a circuit description for a unitary U, so you can run it, to map $|\psi\rangle$ to $U|\psi\rangle$ for any given state $|\psi\rangle$, or run controlled-U, by controlling on each gate.

Given: a circuit description for a unitary U on H,
a state $|\psi_0\rangle \in H$ **[12]**

Your task is to decide between two distinguishable scenarios.

Task: Decide if:
(1) $U|\psi_0\rangle = |\psi_0\rangle$
(2) $U|\psi_0\rangle = -|\psi_0\rangle$
} promised one of these is true **[13]**

Simply applying U to $|\psi_0\rangle$ won't help, because the result in both cases differs only by a global phase, but fortunately we can also control U on an auxiliary qubit.



$$|\psi'\rangle = (\mathsf{H}\otimes I)\frac{1}{\sqrt{2}}\left(|0\rangle|\psi_0\rangle + |1\rangle U|\psi_0\rangle\right)$$

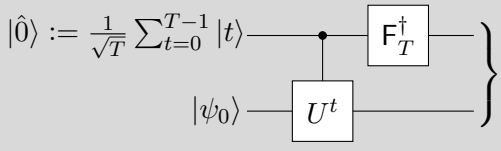
Scenario (1): $|\psi'\rangle = (\mathsf{H}\otimes I)\frac{1}{\sqrt{2}}\left(|0\rangle|\psi_0\rangle + |1\rangle|\psi_0\rangle\right) = \mathsf{H}|+\rangle|\psi_0\rangle = |0\rangle|\psi_0\rangle$
Scenario (2): $|\psi'\rangle = (\mathsf{H}\otimes I)\frac{1}{\sqrt{2}}\left(|0\rangle|\psi_0\rangle - |1\rangle|\psi_0\rangle\right) = \mathsf{H}|-\rangle|\psi_0\rangle = |1\rangle|\psi_0\rangle$ **[14]**

So we can perfectly distinguish these by measuring the first qubit. Kitaev's phase estimation [Kit96] generalizes this idea.

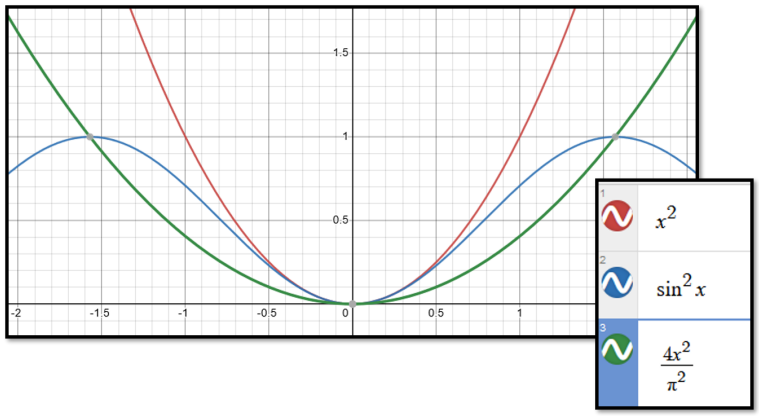
Suppose $U|\psi_0\rangle = e^{i\theta}|\psi_0\rangle$ for some $\theta \in (-\pi, \pi]$.
Task: Estimate θ to ℓ bits of precision, by outputting some $\tilde{\theta}$ such that $|\tilde{\theta} - \theta| \le \frac{\pi}{2^\ell}$. **[15]**

To this end, let $T = 2^\ell$, and consider running the following circuit, where we let F_T denote the quantum Fourier transform $\mathsf{QFT}_{\mathbb{Z}_T}$ over the cyclic group of order T, and a controlled U^t on a wire of dimension T indicates that U^t is applied controlled on the value t on the control wire.

Algorithm 4. Phase Estimation



$$|\psi'\rangle = \frac{1}{\sqrt{T}}(\mathsf{F}_T^\dagger \otimes I)\sum_{t=0}^{T-1}|t\rangle U^t|\psi_0\rangle = \mathsf{F}_T^\dagger \underbrace{\left(\frac{1}{\sqrt{T}}\sum_{t=0}^{T-1} e^{it\theta}|t\rangle\right)}_{(\star)}|\psi_0\rangle$$



Fact 2.2.1. *For any $x \in [-\pi/2, \pi/2]$, $\frac{4x^2}{\pi^2} \leq \sin^2 x$. For any $x \in \mathbb{R}$, $\sin^2 x \leq x^2$.*

The complexity of Algorithm 4 depends on the complexity of preparing $|\psi_0\rangle$, which must be done once, and the complexity of implementing U, which must be done $O(T)$ times (controlled). Additional overhead, such as implementing F_T, is polylogarithmic in T.

> For simplicity, suppose $\theta = \frac{2\pi z}{T}$ for some $z \in \{-T/2+1, \ldots, 0, \ldots, T/2\}$.
> Then $(\star) = \frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} e^{2\pi i \frac{zt}{T}} |t\rangle = \mathsf{F}_T |z\rangle =: |\hat{z}\rangle$ is a Fourier basis vector.

So we can perfectly distinguish these cases by measuring the first register in the Fourier basis, which is precisely what we do when we apply $\mathsf{F}_T^\dagger$ and then measure (in the computational basis).

Exercise 2.2.1. *Prove that for any positive integer T and $x \in \mathbb{R} \setminus \{0\}$, $\left|\sum_{t=0}^{T-1} e^{ixt}\right|^2 = \frac{\sin^2 \frac{Tx}{2}}{\sin^2 \frac{x}{2}}$.*

Exercise 2.2.2. *Let θ be an arbitrary angle in $(-\pi, \pi]$, and suppose $U|\psi_0\rangle = e^{i\theta}|\psi_0\rangle$. Let z be such that $\left|\theta - 2\pi z/2^\ell\right| \leq \frac{\pi}{2^\ell}$. Show that the probability that Algorithm 4 outputs z, upon measuring the first register, is at least $4/\pi^2$. You may use Fact 2.2.1.*

While the above exercise shows that it is reasonably likely to measure the best possible ℓ-bit estimate of θ, since $4/\pi^2 < 1/2$, it does not rule out that there is another terrible estimate that is just as likely to occur, so this analysis is not really sufficient. The following exercise gives an improved analysis.

Exercise 2.2.3. *Let θ be an arbitrary angle in $(-\pi, \pi]$, and suppose $U|\psi_0\rangle = e^{i\theta}|\psi_0\rangle$. Let $k \in \mathbb{Z}$ such that $k \geq 2$, and let $\tilde{\theta} = \frac{2\pi z}{T}$ where z is the outcome of measuring the first register in Algorithm 4. Show that the probability that*

$$\left|\tilde{\theta} - \theta\right| \leq \frac{2\pi k}{T}$$

is at least $1 - \frac{1}{2(k-1)}$. Hint: You may use the fact that $\sum_{z=k}^{\infty} \frac{1}{z^2} \leq \frac{1}{k-1}$. It may be helpful to start by arguing that we can assume that the closest angle in $\{2\pi z/T : z \in \{-(T-1)/2, \ldots, T/2\}\}$ to θ is 0.

The above analysis is sufficient: taking $k > 2$ guarantees that we have a pretty good estimate of θ with probability at least $3/4$. By repeating this $O(\log \frac{1}{\varepsilon})$ times and taking the *median* of all outcomes, we can increase the probabilty of getting a good estimate to $1 - \varepsilon$.

Exercise 2.2.4. *In the above reasoning, why is it not sufficient to take the mode of all estimates?*

2.2.1 Application to OR and approximate counting

We will show how to use phase estimation to solve the following problem.

Problem: $\textsc{Approx}_{N,\varepsilon}$

Input: $x \in \{0,1\}^N$
Output: $\tilde{t} \in \{0,\dots,N\}$ such that $\left|\tilde{t} - t\right| \le \varepsilon t$ where $t = |x|$ is the Hamming weight of x.

For simplicity, assume $N = 2^n$ for some integer n. We will solve this problem by doing phase estimation on $\mathcal{G}$, the *Grover iterate*, which you used in [dW19, Section 7.2] to build up Grover's algorithm. Recall that $\mathcal{G}$ is defined

$$\mathcal{G} = \underbrace{\mathsf{H}^{\otimes n} R_0 \mathsf{H}^{\otimes n}}_{=:R_\pi} \mathcal{O}_x^{\pm}, \tag{2.6}$$

where

$$R_0 =: |0\rangle\langle 0| - \sum_{\substack{i \in \{0,1\}^n: \\ i \neq 0}} |i\rangle\langle i|.$$

We will assume $|x| \le N/2$, as otherwise we can estimate $N - |x|$ instead.

Exercise 2.2.5. *Let $t = |x|$, and let $\theta \in [0, \pi/2]$ be such that $\sin^2\theta = t/N$. Suppose $t > 0$, and define*

$$|\psi_\pm\rangle := \frac{1}{\sqrt{2}}\left(\sqrt{\frac{N}{t}}\underbrace{\frac{1}{\sqrt{N}}\sum_{i:x_i=1}|i\rangle}_{=:|\psi_{good}\rangle} \pm i\sqrt{\frac{N}{N-t}}\underbrace{\frac{1}{\sqrt{N}}\sum_{i:x_i=0}|i\rangle}_{=:|\psi_{bad}\rangle}\right).$$

Show that $\mathcal{G}|\psi_\pm\rangle = e^{\pm i2\theta}|\psi_\pm\rangle$. Hint: Go to the wikipedia page on trigonometric identities.

Exercise 2.2.6. *Let $|\pi\rangle := \sum_{i\in\{0,1\}^n} \frac{1}{\sqrt{N}}|i\rangle$. Show that $|\pi\rangle \in \operatorname{span}\{|\psi_+\rangle, |\psi_-\rangle\}$.*

With the above two exercises, we can estimate t using the following algorithm.

Algorithm 5. Approximate Counting by Phase Estimation

1. `Apply Phase Estimation (Algorithm 4) with` $|\psi_0\rangle = |\pi\rangle$, $T = 2^\ell$, `and` $U = \mathcal{G}$.
2. `Meaure the first register to obtain some` $z \in \{0,\dots,T-1\}$, `and output` $\tilde{t} = N\sin^2\left(\frac{z\pi}{T}\right)$.

Lemma 2.2.2. *Let $\varepsilon \in (0,1)$, and let $t = |x|$. Suppose $0 < t \le \frac{N}{2}$, and T is such that $T \ge \frac{16\pi}{\varepsilon}\sqrt{\frac{N}{t}}$. Then with probability at least $\frac{3}{4}$, the output of Algorithm 5 satisfies $\left|\tilde{t} - t\right| \le \varepsilon t$.*

Proof. First, note that since $|\pi\rangle \in \operatorname{span}\{|\psi_+\rangle, |\psi_-\rangle\}$, and these two states are orthogonal, we can analyze the case where each of these two states is used as input, and extrapolate by linearity. Depending on which state is used, the phase is either 2θ, or -2θ, so we can analyze both cases together.

Let $\tilde{\theta} = 2\pi z/T$, where z is the measurement outcome, so $\tilde{t} = N\sin^2\frac{\tilde{\theta}}{2}$. By Exercise 2.2.3, using $k = 3$, with probabiliy at least $\frac{3}{4}$, we have

$$\begin{aligned}
\left|\tilde{\theta} - 2\theta\right| \le \frac{6\pi}{T} \quad &\text{or} \quad \left|\tilde{\theta} - (-2\theta)\right| \le \frac{6\pi}{T} \\
-\frac{6\pi}{T} \le \tilde{\theta} - 2\theta \le \frac{6\pi}{T} \quad &\text{or} \quad -\frac{6\pi}{T} \le \tilde{\theta} + 2\theta \le \frac{6\pi}{T} \\
\theta - \frac{3\pi}{T} \le \frac{\tilde{\theta}}{2} \le \theta + \frac{3\pi}{T} \quad &\text{or} \quad \theta - \frac{3\pi}{T} \le -\frac{\tilde{\theta}}{2} \le \theta + \frac{3\pi}{T}.
\end{aligned}$$

Assume one of these holds. First, note that if $\theta - \frac{3\pi}{T} \le \frac{\tilde\theta}{2}$, we must have $\tilde\theta > 0$ by $T > 3\pi\sqrt{N/t}$; and similarly, when $\theta - \frac{3\pi}{T} \le -\frac{\tilde\theta}{2}$, we must have $\tilde\theta < 0$. Thus, in either case, we have:

$$\theta - \frac{3\pi}{T} \le \left|\frac{\tilde\theta}{2}\right| \le \theta + \frac{3\pi}{T}. \tag{2.7}$$

Note also that since $\theta \in [0, \pi/2]$, and so $\sin\theta \ge \frac{2}{\pi}\theta$, we have

$$\theta + \frac{3\pi}{T} \le \frac{\pi}{2}\sin\theta + \frac{3\pi}{16\pi}\varepsilon\sqrt{\frac{t}{N}} = \frac{\pi}{2}\sqrt{\frac{t}{N}} + \frac{3}{16}\varepsilon\sqrt{\frac{t}{N}} =\le \frac{\pi}{2}\frac{1}{\sqrt{2}} + \frac{3}{16}\frac{1}{\sqrt{2}} < \frac{\pi}{2}.$$

From here, since sin is an increasing function on $[0, \pi/2]$ using $\sin|x| = |\sin x|$,

$$\begin{aligned}
\sin\left|\frac{\tilde\theta}{2}\right| &\le \sin\left(\theta + \frac{3\pi}{T}\right)\\
\left|\sin\frac{\tilde\theta}{2}\right| &\le \sin\theta\cos\left(\frac{3\pi}{T}\right) + \cos\theta\sin\left(\frac{3\pi}{T}\right)\\
\sqrt{\frac{\tilde t}{N}} &\le \sqrt{\frac{t}{N}} + \frac{3\pi}{T} \le \sqrt{\frac{t}{N}} + \frac{3\pi}{16\pi}\varepsilon\sqrt{\frac{t}{N}},
\end{aligned}$$

where we used $\cos x \le 1$ and $\sin x \le x$ for $x \ge 0$. Using the inequality $(1+\delta)^2 \le 1 + 3\delta$ for $\delta \le 1$, we have:

$$\begin{aligned}
\sqrt{\frac{\tilde t}{N}} &\le \sqrt{\frac{t}{N}}\left(1 + \frac{3}{16}\varepsilon\right)\\
\frac{\tilde t}{N} &\le \frac{t}{N}\left(1 + \frac{9}{16}\varepsilon\right) \le \frac{t}{N}(1+\varepsilon).
\end{aligned} \tag{2.8}$$

Similarly, since $\sin\theta \le \theta$ we have

$$\theta - \frac{3\pi}{T} \ge \sin\theta - \frac{3\pi}{16\pi}\varepsilon\sqrt{\frac{t}{N}} = \left(1 - \frac{3\varepsilon}{16}\right)\sqrt{\frac{t}{N}} \ge 0,$$

and from here, since sin is increasing on $[0, \pi/2]$, (2.7) implies:

$$\begin{aligned}
\sin\left|\frac{\tilde\theta}{2}\right| &\ge \sin\left(\theta - \frac{3\pi}{T}\right) = \sin\theta\cos\frac{3\pi}{T} - \cos\theta\sin\frac{3\pi}{T}\\
\sqrt{\frac{\tilde t}{N}} &\ge \sin\theta\sqrt{1 - \sin^2\frac{3\pi}{T}} - \sin\frac{3\pi}{T} \ge \sqrt{\frac{t}{N}}\sqrt{1 - \frac{9\pi^2}{T^2}} - \frac{3\pi}{T},
\end{aligned}$$

where we used $\cos \le 1$ and $\sin x \le x$. Plugging in T, and using the inequalities $\sqrt{1-x} \ge 1 - x$ when $x \in [0,1]$, and $(1-x)^2 \ge 1 - 2x$, we get

$$\begin{aligned}
\sqrt{\frac{\tilde t}{N}} &\ge \sqrt{\frac{t}{N}}\left(1 - \frac{9\pi^2}{(16\pi)^2}\varepsilon^2\frac{t}{N}\right) - \frac{3\pi}{16\pi}\varepsilon\sqrt{\frac{t}{N}}\\
\frac{\tilde t}{N} &\ge \frac{t}{N}\left(1 - \frac{9}{64}\varepsilon^2\frac{t}{N} - \frac{3\varepsilon}{16}\right)^2 \ge \frac{t}{N}\left(1 - \frac{9}{64}\frac{\varepsilon}{2} - \frac{3\varepsilon}{16}\right)^2 \ge \frac{t}{N}\left(1 - \frac{\varepsilon}{2}\right)^2 \ge \frac{t}{N}(1-\varepsilon).
\end{aligned}$$

Combining this with (2.8), we get $t(1-\varepsilon) \le \tilde t \le t(1+\varepsilon)$, implying the claim. □

The decision version of Search, which is also called Or, reduces to $\textsc{Approx}_{n,0}$, however, our analysis skips the case where $t = 0$. The following exercise covers this case.

Exercise 2.2.7. *1. Show that if $t = 0$, $|\pi\rangle$ is an e^{i0}-eigenvector of $\mathcal{G}$.*

2. Let $\mathrm{OR}_N(x) = 1$ if and only if $|x| > 0$, where $x \in \{0,1\}^N$. Show how to solve OR_N with one-sided error ε, for some $\varepsilon < 1$, using one call to Algorithm 5, with an appropriate choice of T. What is the query complexity of your algorithm?

2.2.2 Phase Estimation Algorithms for Products of Reflections

It turns out that many, perhaps most, applications of phase estimation only rely on the ability to distinguish the case $\theta = 0$ from the case $\theta \neq 0$. We will show how to use Algorithm 4 to decide if $\theta = 0$ or $\theta \neq 0$ in the special case where U is a product of two reflections. More precisely, we will distinguish between the case when $|\psi_0\rangle$ is close to a 1-eigenstate of U, and the case where it has no overlap with any 1-eigenvectors, or even $e^{i\theta}$-eigenvectors for small θ. The complexity of this will naturally depend on what we mean by "close" and "small". These will be determined by the relationship between $|\psi_0\rangle$ and the two reflections that make up U.

Consider the probability of measuring 0 in the first register of Algorithm 4, p_0, for the case $U|\psi_0\rangle = e^{i\theta}|\psi_0\rangle$ for *any* $\theta \in (-\pi, \pi]$. In general, $|\psi_0\rangle$ need not actually be an eigenvector of U, but it will always be a linear combination of them.

Probability of measuring 0 in Algorithm 4:

$$p_0 = \left\|(\langle 0| \otimes I)|\psi'\rangle\right\|^2 = \left\|(\langle 0|\mathsf{F}_T^\dagger \otimes I)\sum_{t=0}^{T-1}\frac{1}{\sqrt{T}}e^{i\theta t}|t\rangle|\psi_0\rangle\right\|^2 = \left\|\left(\sum_{t=0}^{T-1}\frac{1}{\sqrt{T}}\langle t|\right)\sum_{t=0}^{T-1}\frac{1}{\sqrt{T}}e^{i\theta t}|t\rangle\right\|^2$$
$$= \left\|\frac{1}{T}\sum_{t=0}^{T-1}e^{i\theta t}\right\|^2 = \frac{1}{T^2}\left|\sum_{t=0}^{T-1}e^{i\theta t}\right|^2 = \begin{cases} 1 & \text{if } \theta = 0 \\ \frac{1}{T^2}\left|\frac{1-e^{i\theta T}}{1-e^{i\theta}}\right|^2 = \frac{1}{T^2}\frac{4\sin^2\frac{T\theta}{2}}{4\sin^2\frac{\theta}{2}} & \text{if } \theta \neq 0 \end{cases} \tag{2.9}$$

[17]

When $\theta = 0$, $p_0 = 1$, so we measure 0 in the phase register with probability 1. When $\theta \neq 0$, we hope that p_0 is small, as long as θ is not too close to 0. The expression $\frac{\sin\frac{T\theta}{2}}{\sin\frac{\theta}{2}} = \mathsf{U}_{T-1}(\cos\theta)$, where U_{T-1} is the $(T-1)^{\text{th}}$ Chebyshev polynomial of the second kind. We will not use this fact at the moment, but when strange-looking expressions have a name, it oftens means that very smart people have thought a lot about how to make sense of them, which is generally quite useful. For now though, to upper bound p_0, it will be enough to use the very useful fact that $\sin^2 x \approx x^2$ when $x \in [-\pi/2, \pi/2]$. More precisely, refer to Fact 2.2.1.

So if $\theta \neq 0$, $p_0 \leq \frac{1}{T^2}\frac{1}{\sin^2\frac{\theta}{2}} \leq \frac{1}{T^2}\frac{1}{\frac{4}{\pi^2}(\frac{\theta}{2})^2} = \frac{\pi^2}{(T\theta)^2}$. This is at most any choice of $1/c^2$ if $\theta \geq \frac{\pi^2}{cT}$.

[18]

So Algorithm 4 may confuse some small phases θ with 0, but it's less likely as θ or T gets larger.

We have been assuming that $|\psi_0\rangle$ is an eigenvector of U, but that needn't be the case. Let us decompose U in terms of projectors onto its eigenspaces $\{\Pi_j\}_{j\in J}$ for some finite set of labels J, with $e^{i\theta_j}$ being the corresponding eigenvalues, for $\theta_j \in (-\pi, \pi]$:

$$U = \sum_{j\in J} e^{i\theta_j}\Pi_j \qquad \text{for } \{\theta_j : j \in J\} \subset (-\pi, \pi]. \tag{2.10}$$

Then $|\psi_0\rangle = \sum_{j\in J} \Pi_j|\psi_0\rangle$, so from (2.9):

$$p_0 = \sum_{j\in J:\theta_j\neq 0} \frac{1}{T^2}\frac{\sin^2\frac{T\theta_j}{2}}{\sin^2\frac{\theta_j}{2}} \|\Pi_j|\psi_0\rangle\|^2 + \|\Lambda_0|\psi_0\rangle\|^2 \tag{2.11}$$

where we use Λ_0 to denote the projector onto the $(+1)$-eigenspace of U (so it's either the Π_j for which $\theta_j = 0$, or if there is no such j, it's 0).

As promised, we will constrain U to be a product of two reflections.

Let $\Psi_\mathcal{A}, \Psi_\mathcal{B} \subset H$ be finite sets of vectors, $\mathcal{A} = \text{span}\{\Psi_\mathcal{A}\}$, $\mathcal{B} = \text{span}\{\Psi_\mathcal{B}\}$ subspaces of H, and

$$U = (2\Pi_\mathcal{A} - I)(2\Pi_\mathcal{B} - I)$$

where $\Pi_\mathcal{A}$ and $\Pi_\mathcal{B}$ are the orthogonal projectors onto $\mathcal{A}$ and $\mathcal{B}$.

There are two reasons for considering this special case of U. First, we are assuming we have a description of U that allows us to implement it as a circuit. Suppose $\Psi_\mathcal{A} = \{|\psi_1\rangle, \ldots, |\psi_k\rangle\}$ is a set of *orthonormal* vectors.

Let $U_\mathcal{A}|j\rangle|0\rangle = |\psi_j\rangle$ for all $j \in [k]$, in which case, we say $U_\mathcal{A}$ *generates* $\Psi_\mathcal{A}$. Then

$$(2\Pi_\mathcal{A} - I) = U_\mathcal{A} \underbrace{\left(2\sum_{j=1}^{k} |j\rangle\langle j| \otimes |0\rangle\langle 0| - I\right)}_{\text{probably easy}} U_\mathcal{A}^\dagger.$$

That is, implementing U is as easy as *generating* $\Psi_\mathcal{A}$ and $\Psi_\mathcal{B}$. This is a generalization of how the reflection R_π is implemented in Grover's algorithm (see (2.6)), where $\mathsf{H}^{\otimes n}$ generates $|\pi\rangle$.

The second reason for considering this special case is that the additional structure tells us something about the eigenspaces of U:

We have that the $(+1)$-eigenspace of U is $(\mathcal{A}\cap\mathcal{B}) \oplus (\mathcal{A}^\perp \cap \mathcal{B}^\perp)$.
We additionally assume $|\psi_0\rangle \in \mathcal{B}^\perp$, so $\Lambda_0|\psi_0\rangle \in \mathcal{A}^\perp \cap \mathcal{B}^\perp = (\mathcal{A}+\mathcal{B})^\perp$.

Exercise 2.2.8. *Show that for any two subspaces of H, $\mathcal{A}^\perp \cap \mathcal{B}^\perp = (\mathcal{A}+\mathcal{B})^\perp$. Note that when we fix the space H, $\mathcal{A}^\perp$ is defined as those vectors* in *H that are orthogonal to everything in $\mathcal{A}$, and similarly for $\mathcal{B}^\perp$.*

So to summarize, the parameters of a 0-Phase Estimation Algorithm are (for an implicit input x):

Definition 2.2.3 (Parameters of a 0-Phase Estimation Algorithm)**.** *Fix a finite-dimensional complex inner product space H, a unit vector $|\psi_0\rangle \in H$, and sets of vectors $\Psi_\mathcal{A}, \Psi_\mathcal{B} \subset H$. We further assume that $|\psi_0\rangle$ is orthogonal to every vector in $\Psi_\mathcal{B}$. Let $\Pi_\mathcal{A}$ be the orthogonal projector onto $\mathcal{A} = \text{span}\{\Psi_\mathcal{A}\}$, and similarly for $\Pi_\mathcal{B}$.*

Let us be more specific in qualifying the two scenarios that we will use Algorithm 4 to distinguish.

Positive Case: $\|\Lambda_0|\psi_0\rangle\|^2 \neq 0 \Leftrightarrow \exists$ a positive witness **[23]**

Definition 2.2.4 (Positive Witness)**.** *A* positive witness *(for $(H, |\psi_0\rangle, \Psi_\mathcal{A}, \Psi_\mathcal{B})$) is a vector $|w\rangle \in H$ such that $\langle\psi_0|w\rangle \neq 0$ and $|w\rangle \in \mathcal{A}^\perp \cap \mathcal{B}^\perp$.*

That is, a positive witness $|w\rangle$ is exactly a non-zero component of $|\psi_0\rangle$ in the $(+1)$-eigenspace of U. Demonstrating the existence of a positive witness proves that we are in the positive case[2]. Moreover, the smaller the witness relative to its overlap with $|\psi_0\rangle$, the "more positive" the case.

Lemma 2.2.5 (Positive Lemma)**.** *If $|w\rangle$ is a positive witness, then $\|\Lambda_0|\psi_0\rangle\|^2 \geq \frac{|\langle w|\psi_0\rangle|^2}{\||w\rangle\|^2}$.*

Proof.

$$\|\Lambda_0|\psi_0\rangle\|^2 = \left\| \frac{|w\rangle\langle w|}{\||w\rangle\|^2}|\psi_0\rangle + \left(\Lambda_0 - \frac{|w\rangle\langle w|}{\||w\rangle\|^2}\right)|\psi_0\rangle \right\|^2 \geq \left\| \frac{|w\rangle\langle w|\psi_0\rangle}{\||w\rangle\|^2} \right\|^2 = \frac{|\langle w|\psi_0\rangle|^2}{\||w\rangle\|^2}. \qquad \square$$

So $\frac{|\langle w|\psi_0\rangle|}{\||w\rangle\|}$ is bigger the more of $|\psi_0\rangle$ is in the $(+1)$-eigenspace. If $|w\rangle$ is the projection of $|\psi_0\rangle$ onto the $(+1)$-eigenspace, then the proof of Lemma 2.2.5 is tight.

Negative Case: $\|\Lambda_0|\psi_0\rangle\|^2 = 0 \Leftrightarrow \exists$ a negative witness **[24]**

Definition 2.2.6 (Negative Witness)**.** *A* negative witness *(for $(H, |\psi_0\rangle, \Psi_\mathcal{A}, \Psi_\mathcal{B})$) is a pair of vectors $|w_\mathcal{A}\rangle, |w_\mathcal{B}\rangle \in H$ such that $|w_\mathcal{A}\rangle + |w_\mathcal{B}\rangle = |\psi_0\rangle$, $|w_\mathcal{A}\rangle \in \mathcal{A}$ and $|w_\mathcal{B}\rangle \in \mathcal{B}$.*

That is there is a negative witness exactly when $|\psi_0\rangle \in \mathcal{A} + \mathcal{B} = (\mathcal{A}^\perp \cap \mathcal{B}^\perp)^\perp$. Since $|\psi_0\rangle \in \mathcal{B}^\perp \subseteq (\mathcal{A} \cap \mathcal{B})^\perp$, this happens if and only if $\Lambda_0|\psi_0\rangle = 0$.

Let $\Lambda_\Theta = \sum_{j \in J : |\theta_j| \leq \Theta} \Pi_j$ for $\Theta \in [0, \pi)$. **[25]**

Lemma 2.2.7 (Negative Lemma)**.** *If $|w_\mathcal{A}\rangle, |w_\mathcal{B}\rangle$ is a negative witness, then for any $\Theta \in [0, \pi)$, $\|\Lambda_\Theta|\psi_0\rangle\|^2 \leq \frac{\Theta^2}{4}\||w_\mathcal{A}\rangle\|^2$.*

Consider this statement for $\Theta = 0$. That tells us that if there is a negative witness, then there is no component of $|\psi_0\rangle$ in the $(+1)$-eigenspace of U, which we argued above. This fact on its own is not enough for an algorithm to distinguish the negative and positive case, because it still leaves the possibility that $|\psi_0\rangle$ has high overlap with $e^{i\theta}$-eigenspaces for tiny θ that are computationally indistinguishable from 0. That's why we need a statement in terms of the parameter Θ.

For intuition about Lemma 2.2.7, it is helpful to consider the extreme case where $|\psi_0\rangle \in \mathcal{B}^\perp \cap \mathcal{A}$, which is in the (-1)-eigenspace of U. This is as far as possible from the positive case, in which $|\psi_0\rangle$ is in the $(+1)$-eigenspace. In this case, $|w_\mathcal{A}\rangle = |\psi_0\rangle, |w_\mathcal{B}\rangle = 0$ is a negative witness, with $\||w_\mathcal{A}\rangle\| = 1$. As $|\psi_0\rangle$ gets further from $\mathcal{A}$, we require larger $|w_\mathcal{A}\rangle$ (and $|w_\mathcal{B}\rangle$) to add up to $|\psi_0\rangle$. See Figure 2.3.

To prove Lemma 2.2.7, we need the following lemma.

Lemma 2.2.8 (Effective Spectral Gap Lemma)**.** *Let $\Theta \in [0, \pi)$, and let Λ_Θ be the orthogonal projector onto the $e^{i\theta}$-eigenspaces of $U = (2\Pi_\mathcal{A} - I)(2\Pi_\mathcal{B} - I)$ for $|\theta| \leq \Theta$. If $|\psi_\mathcal{A}\rangle \in \mathcal{A}$, then $\|\Lambda_\Theta(I - \Pi_\mathcal{B})|\psi_\mathcal{A}\rangle\|^2 \leq \frac{\Theta^2}{4}\||\psi_\mathcal{A}\rangle\|^2$.*

[2] This is something we will do when analysing a 0-phase estimation algorithm. The algorithm itself does not produce a positive witness.

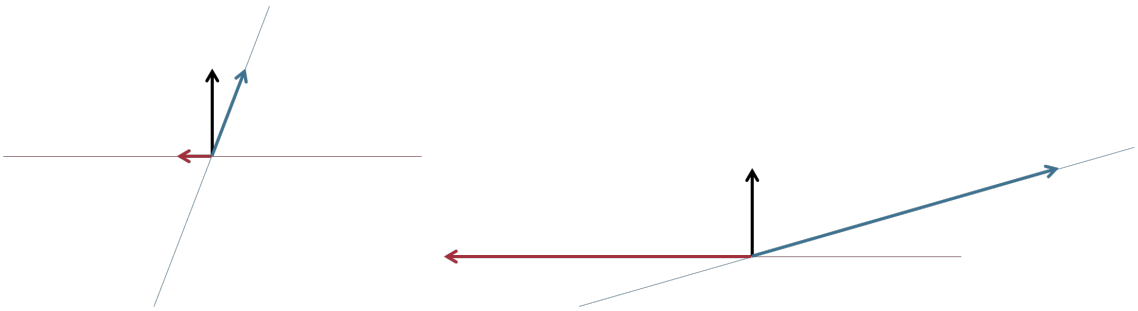


Figure 2.3: We see two examples of the plane spanning $|w_{\mathcal{A}}\rangle$ (blue) and $|w_{\mathcal{B}}\rangle$ (red), with $|\psi_0\rangle = |w_{\mathcal{A}}\rangle + |w_{\mathcal{B}}\rangle$ shown in black. We always have $|\psi_0\rangle \in \mathcal{B}^\perp$ orthogonal to $|w_{\mathcal{B}}\rangle$. On the left we see a case where $|\psi_0\rangle$ is almost in $\mathcal{A}$, meaning it's almost in $\mathcal{A} \cap \mathcal{B}^\perp$ which is in the (-1)-eigenspace of U, and $|w_{\mathcal{A}}\rangle$ is very small. On the right we see a case where $|\psi_0\rangle$ is far from $\mathcal{A}$ (or at least, far from $\text{span}\{|w_{\mathcal{A}}\rangle\}$), and so $|w_{\mathcal{A}}\rangle$ is much larger.

Exercise 2.2.9. *Prove the Effective Spectral Gap Lemma. To do so, first express $(I - U^\dagger)|\psi_{\mathcal{A}}\rangle$ in terms of $\Pi_{\mathcal{B}}$ (it turns out not to depend on $\Pi_{\mathcal{A}}$). Then use a decomposition of $(I - U^\dagger)$ as a linear combination of the Π_j (see (2.10)).*

Proof of Lemma 2.2.7. Since $|w_{\mathcal{A}}\rangle \in \mathcal{A}$, by the Effective Spectral Gap Lemma:

$$\|\Lambda_\Theta(I - \Pi_{\mathcal{B}})|w_{\mathcal{A}}\rangle\|^2 \leq \frac{\Theta^2}{4} \||w_{\mathcal{A}}\rangle\|^2$$

$$\|\Lambda_\Theta \underbrace{(I - \Pi_{\mathcal{B}})|\psi_0\rangle}_{=|\psi_0\rangle \text{ since } |\psi_0\rangle \in \mathcal{B}^\perp} - \Lambda_\Theta \underbrace{(I - \Pi_{\mathcal{B}})|w_{\mathcal{B}}\rangle}_{=0 \text{ since } |w_{\mathcal{B}}\rangle \in \mathcal{B}} \|^2 \leq \frac{\Theta^2}{4} \||w_{\mathcal{A}}\rangle\|^2 \qquad \text{since } |\psi_0\rangle = |w_{\mathcal{A}}\rangle + |w_{\mathcal{B}}\rangle$$

$$\|\Lambda_\Theta|\psi_0\rangle\|^2 \leq \frac{\Theta^2}{4} \||w_{\mathcal{A}}\rangle\|^2 . \qquad \square$$

Now we can combine the positive and negative lemmas into a main theorem.

Theorem 2.2.9 (0-Phase Estimation on a Product of Reflections)**.** *Suppose we can generate $|\psi_0\rangle$ in cost S and implement U in cost A. Let c_+ be a constant, and $\mathcal{C}_-$ a positive real number (which may scale with some implicit input) such that exactly one of the following is promised to hold:*

Positive Condition: *There is a positive witness $|w\rangle$ s.t. $\frac{|\langle w|\psi_0\rangle|^2}{\||w\rangle\|^2} \geq \frac{1}{c_+}$.*

Negative Condition: *There is a negative witness $|w_{\mathcal{A}}\rangle, |w_{\mathcal{B}}\rangle$ s.t. $\||w_{\mathcal{A}}\rangle\|^2 \leq \mathcal{C}_-$.*

Then running Algorithm 4 with $T = \pi^2 c_+ \sqrt{\mathcal{C}_-}$, measuring the first register, and outputting "positive" if and only if the result is 0, decides between these two cases with bounded error in cost $O(\mathsf{S} + \sqrt{\mathcal{C}_-}\mathsf{A})$.

Proof. For the **Positive Case**, suppose the Positive Condition holds. Then from (2.11), the probability of measuring 0 satisfies:

$$p_0 \geq \|\Lambda_0|\psi_0\rangle\|^2 \geq \frac{|\langle w|\psi_0\rangle|^2}{\||w\rangle\|^2} \geq \frac{1}{c_+}$$

by the Positive Condition and the Positive Lemma (Lemma 2.2.5).

For the **Negative Case**, suppose the Negative Condition holds. Then from (2.11), letting $\Theta =$

$\frac{2}{\pi\sqrt{c_+\mathcal{C}_-}}$, we have:

$$\begin{aligned}
p_0 &= \sum_{j:|\theta_j|>\Theta} \frac{1}{T^2}\frac{\sin^2\left(\frac{\theta_j T}{2}\right)}{\sin^2\left(\frac{\theta_j}{2}\right)}\|\Pi_j|\psi_0\rangle\|^2 + \sum_{j:0<|\theta_j|\le\Theta} \frac{1}{T^2}\frac{\sin^2\left(\frac{\theta_j T}{2}\right)}{\sin^2\left(\frac{\theta_j}{2}\right)}\|\Pi_j|\psi_0\rangle\|^2 + \|\Lambda_0|\psi_0\rangle\|^2 \\
&\le \sum_{j:|\theta_j|>\Theta} \frac{1}{T^2}\frac{1}{\left(\frac{\theta_j}{\pi}\right)^2}\|\Pi_j|\psi_0\rangle\|^2 + \sum_{j:0<|\theta_j|\le\Theta} \frac{1}{T^2}\frac{\left(\frac{\theta_j T}{2}\right)^2}{\left(\frac{\theta_j}{\pi}\right)^2}\|\Pi_j|\psi_0\rangle\|^2 + \|\Lambda_0|\psi_0\rangle\|^2 \\
&< \frac{1}{T^2}\frac{\pi^2}{\Theta^2}\sum_{j:|\theta_j|>\Theta}\|\Pi_j|\psi_0\rangle\|^2 + \frac{\pi^2}{4}\sum_{j:0<|\theta_j|\le\Theta}\|\Pi_j|\psi_0\rangle\|^2 + \frac{\pi^2}{4}\|\Lambda_0|\psi_0\rangle\|^2 \\
&\le \frac{1}{T^2}\frac{\pi^2}{\Theta^2} + \frac{\pi^2}{4}\|\Lambda_\Theta|\psi_0\rangle\|^2 \\
&\le \frac{1}{T^2}\frac{\pi^2}{\Theta^2} + \frac{\pi^2}{4}\frac{\Theta^2}{4}\||w_{\mathcal{A}}\rangle\|^2 && \text{Lemma 2.2.7} \\
&\le \frac{1}{T^2}\frac{\pi^2}{\Theta^2} + \frac{\pi^2}{4}\frac{\Theta^2}{4}\mathcal{C}_- && \text{by Neg. Cond.}
\end{aligned}$$

Plugging in $\Theta = \frac{2}{\pi\sqrt{c_+\mathcal{C}_-}}$ and $T = \pi^2 c_+\sqrt{\mathcal{C}_-}$, this becomes:

$$p_0 \le \frac{1}{\pi^4 c_+^2\mathcal{C}_-}\frac{\pi^2}{4}\pi^2 c_+\mathcal{C}_- + \frac{\pi^2}{16}\frac{4}{\pi^2 c_+\mathcal{C}_-}\mathcal{C}_- = \frac{1}{4c_+} + \frac{1}{4c_+} = \frac{1}{2}\frac{1}{c_+}.$$

So in the positive case, $p_0 \ge \frac{1}{c_+}$, and in the negative case, $p_0 \le \frac{1}{2}\frac{1}{c_+}$. Since c_+ is constant, we can distinguish these with bounded error in $O(1)$ repetitions of Algorithm 4, since $\frac{1}{c_+} - \frac{1}{2c_+} = \Omega(1)$. The proof is completed by verifying that the full algorithm has the claimed cost. □

Further Directions Sometimes we want to distinguish between the case when $|\psi_0\rangle$ is *almost* in $\mathcal{A}+\mathcal{B}$ vs. far from being in $\mathcal{A}+\mathcal{B}$. This can be handled by an approximate version of Theorem 2.2.9, which relaxes the condition on both positive and negative witnesses to allow for some deviation [JZ23].

Example: Promise search

For an integer $t > 0$, possibly a function of N, the decision version of t-promise search is defined:

Problem: $\textsc{Or}_{N,t}$

Input $x \in \{0,1\}^N$ such that $|x| = 0$ or $|x| \ge t$
Output 1 if $|x| \ge t$, 0 if $|x| = 0$.

This problem is already solved by Algorithm 5 with a correct choice of T, but in this section, we will also describe a 0-phase estimation algorithm (using Theorem 2.2.9) for this problem.

Exercise 2.2.10. *Show that one call to Algorithm 5, with a correct choice of T, can be used to decide $\textsc{Or}_{N,t}$ with one-sided error. What is the complexity of your algorithm? You may refer to statements proven in Lemma 2.2.2 without copying their derivations.*

Fix the following phase estimation algorithm parameters:

$$
\begin{aligned}
H &= \mathrm{span}\{|i\rangle : i \in [N]\} \oplus \mathrm{span}\{|0\rangle\} \\
\Psi_{\mathcal{A}} &= \left\{ |\psi_{\mathcal{A}}\rangle := \sqrt{t}|0\rangle + \sum_{i=1}^{N} |i\rangle \right\} \\
\Psi_{\mathcal{B}} &= \{|i\rangle : x_i = 0\} \\
|\psi_0\rangle &= |0\rangle.
\end{aligned}
$$

We will show that Theorem 2.2.9 with these parameters gives an algorithm that decides $\mathrm{OR}_{N,t}$.

Exercise 2.2.11. *Show that there is a choice of constant $c_+ \in [1, 50]$ such that when $|x| \geq t$, there exists a positive witness $|w\rangle$ such that $\frac{|\langle\psi_0|w\rangle|^2}{\||w\rangle\|^2} \geq \frac{1}{c_+}$.*

Exercise 2.2.12. *Show that there is a choice of $\mathcal{C}_- = O(N/t)$ such that when $|x| = 0$, there exists a negative witness $|w_{\mathcal{A}}\rangle, |w_{\mathcal{B}}\rangle$ such that $\||w_{\mathcal{A}}\rangle\|^2 \leq \mathcal{C}_-$.*

Thus, with the chosen values of $c_+ = O(1)$ and $\mathcal{C}_- = O(N/t)$, our parameters satisfies the conditions of Theorem 2.2.9. That is, when $|x| = 0$, we satisfy the negative condition, and when $|x| \geq t$, we satisfy the positive condition, and so we can distinguish these two conditions with bounded error in cost

$$
O(\mathsf{S} + \sqrt{\mathcal{C}_-}\mathsf{A}) = O\left(\mathsf{S} + \sqrt{\frac{N}{t}}\mathsf{A}\right).
$$

It remains to analyze S and A. First, S is the cost of generating $|\psi_0\rangle = |0\rangle = |0^{\log N}\rangle$, which is 0 queries to x, or $O(\log N)$ total operations (for initializing $\log N$ qubits). A is the cost of implementing $U = (2\Pi_{\mathcal{A}} - I)(2\Pi_{\mathcal{B}} - I)$. If we only care about query complexity, then we can implement $(2\Pi_{\mathcal{A}} - I)$ in 0 queries, and $(2\Pi_{\mathcal{B}} - I)$ is just a single query, so $\mathsf{A} = 1$. If we care about counting all operations, then we can implement $(2\Pi_{\mathcal{A}} - I)$ as follows. First, note that we can implement a unitary V that performs $|0\rangle = |0^{\log N}\rangle \mapsto \sqrt{\frac{t}{t+N}}|0\rangle + \frac{1}{\sqrt{t+N}}\sum_{i=1}^{N}|i\rangle = \frac{1}{\sqrt{t+N}}|\psi_{\mathcal{A}}\rangle$ in $O(\log N)$ gates (Hint: use an auxillary qubit). Then

$$
V(2|0\rangle\langle 0| - I)V^\dagger = 2V|0\rangle\langle 0|V^\dagger - I = 2\frac{|\psi_{\mathcal{A}}\rangle\langle\psi_{\mathcal{A}}|}{\||\psi_{\mathcal{A}}\rangle\|^2} - I = 2\Pi_{\mathcal{A}} - I.
$$

Thus, the cost of deciding $\mathrm{OR}_{N,t}$ with bounded error[3] is $O(\sqrt{\mathcal{C}_-}) = O(\sqrt{N/t})$ queries, and $O(\sqrt{\mathcal{C}_-}\log N) = O(\sqrt{N/t}\log N)$ time (using queries and local gates).

Exercise 2.2.13. *Recall that $\mathsf{Q}(\mathrm{OR}_N) = \Omega(\sqrt{N})$, where Q denotes quantum query complexity, and $\mathrm{OR}_N = \mathrm{OR}_{N,1}$. Show that for any $t \in \{1, \ldots, N\}$, the algorithm for $\mathrm{OR}_{N,t}$ described above has optimal quantum query complexity.*

Example: Amplitude Amplification

Many classical algorithms are based on a subroutine, $\mathcal{P}$, that is only successful in finding a solution with some small probability ε, but where a solution can be recognized[4]. Repeating such a procedure $1/\varepsilon$ times, and checking each time if the output is a solution, will find a solution with probability at least some constant $p > 0$. We can express this formally as a classical algorithm.

[3] As you showed in Exercise 2.2.10, this is also possible with one-sided error, but Theorem 2.2.9 only handles bounded (two-sided) error analyses.

[4] We stress that here ε is a lower bound on the success probability, rather than an upper bound on the failure probability.

Algorithm 6. Brute Force Search

1. Repeat $1/\varepsilon$ times:
   (a) Run $\mathcal{P}(x)$ to obtain outcome z
   (b) If z is a solution, output z
2. Output ``no solution found''

The quantum version of this is called *amplitude amplification*, and improves the number of calls to the subroutine quadratically.

Theorem 2.2.10 (Amplitude Amplification [BHMT02])**.** *Let $H = \text{span}\{|z\rangle : z \in \mathcal{Z}\}$ for some finite set $\mathcal{Z}$, and $M \subset \mathcal{Z}$ a* marked set*. Suppose we can implement the following quantum subroutines:*

Solution-sample: *Generate a state $V_{\mathcal{P}}(x)|0\rangle$ for some unitary $V_{\mathcal{P}}(x)$ on H, in cost* A

Check: *For any $z \in \mathcal{Z}$, check if $z \in M$, in cost* C

Let ε be a lower bound on $\|\Pi_M V_{\mathcal{P}}(x)|0\rangle\|^2$, where $\Pi_M = \sum_{z\in M} |z\rangle\langle z|$. Then there is a quantum algorithm that finds $z \in M$ with bounded error in complexity

$$O\left(\frac{1}{\sqrt{\varepsilon}}(\mathsf{A}+\mathsf{C})\right).$$

Above, M represents the set of solutions among the potential solutions, $\mathcal{Z}$. Note that we may also consider the case where $\mathcal{Z} = \mathcal{S} \times \mathcal{W}$ where $\mathcal{S}$ are potential solutions, and $\mathcal{W}$ is some additional workspace. Then M should include all states (s, w) such that s is a solution.

We could, for example, derive $V_{\mathcal{P}}$ from some classical program $\mathcal{P}$ that samples a potential solution, in which case, Theorem 2.2.10 is really just a quadratically faster version of Algorithm 6.

We will use Theorem 2.2.9 to derive a decision version of Theorem 2.2.10, where, instead of *finding* a solution, we will *decide if a solution exists* – we decide whether $V_{\mathcal{P}}$ outputs a solution with probability at least ε, or probability 0 – using $O(\frac{1}{\sqrt{\varepsilon}})$ calls to the unitary $V_{\mathcal{P}}$ and $R_M = 2\sum_{z\in M} |z\rangle\langle z| - I$. Note that R_M can be implemented by checking if $z \in M$: Let $f_M(z) = 1$ if and only of $z \in M$, and compute:

$$|z\rangle|0\rangle \overset{\mathcal{O}_{f_M}}{\mapsto} |z\rangle|f_M(z)\rangle \overset{I\otimes\mathsf{Z}}{\mapsto} (-1)^{f_M(z)}|z\rangle|f_M(z)\rangle \overset{\mathcal{O}_{f_M}}{\mapsto} (-1)^{f_M(z)}|z\rangle|f_M(z)\oplus f_M(z)\rangle = (R_M|z\rangle)\,|0\rangle.$$

We can view $V_{\mathcal{P}}$ as a quantum algorithm that solves a problem with one-sided error, but the error need not be bounded away from 1 by a constant – it can be arbitrarily close to 1. Given such a process, we could classically repeat it $1/\varepsilon$ times, and output 1 if and only if any of the repetitions yields a 1, which would give us a one-sided error algorithm with error bounded by a constant less than 1. Here we will show how to do this with only $O(1/\sqrt{\varepsilon})$ repetitions.

Let $\mathcal{A} = \text{span}\{|z\rangle : z \in M\}$ and $\mathcal{B} = \text{span}\{V_{\mathcal{P}}|0\rangle\}^{\perp}$ so

$$2\Pi_{\mathcal{A}} - I = R_M \quad \text{and} \quad 2\Pi_{\mathcal{B}} - I = -(2V_{\mathcal{P}}|0\rangle\langle 0|V_{\mathcal{P}}^{\dagger} - I).$$

By assumption, we can implement these two reflections in costs C and A, respectively. Let $|\psi_0\rangle = V_{\mathcal{P}}|0\rangle \in \mathcal{B}^{\perp}$. We can generate this state in complexity A.

We will show that Theorem 2.2.9 with these parameters gives an algorithm that decides, with bounded error[5], if $\|\Pi_M V_{\mathcal{P}}|0\rangle\|^2 = 0$ (positive case), or $\|\Pi_M V_{\mathcal{P}}|0\rangle\|^2 \geq \varepsilon$ (negative case).

[5] Theorem 2.2.10 implies that this is also possible with one-sided error, but Theorem 2.2.9 only handles bounded (two-sided) error analyses.

For the **Positive Case**, suppose $\|\Pi_M V_{\mathcal{P}}|0\rangle\|^2 = 0$. Notice that there is only one possibility for a positive witness (up to irrelevant scaling): we need $|w\rangle \in \mathcal{B}^\perp$, so it must be $|w\rangle = V_{\mathcal{P}}|0\rangle$. We must also have $|w\rangle \in \mathcal{A}^\perp = \text{span}\{|z\rangle : z \notin M\}$, which is satisfied precisely when $\|\Pi_M V_{\mathcal{P}}|0\rangle\|^2 = 0$, which we are assuming. Finally, we need $\langle w|\psi_0\rangle \neq 0$, which is satisfied since $|w\rangle = |\psi_0\rangle$. We have:

$$\frac{\||w\rangle\|^2}{|\langle w|\psi_0\rangle|^2} = 1 =: c_+.$$

For the **Negative Case**, suppose $\|\Pi_M V_{\mathcal{P}}|0\rangle\|^2 \geq \varepsilon$, and define

$$|w_{\mathcal{A}}\rangle := \frac{\Pi_M V_{\mathcal{P}}|0\rangle}{\|\Pi_M V_{\mathcal{P}}|0\rangle\|^2} = \frac{\Pi_M|\psi_0\rangle}{\|\Pi_M|\psi_0\rangle\|^2} \in \mathcal{A},$$
$$|w_{\mathcal{B}}\rangle := |\psi_0\rangle - |w_{\mathcal{A}}\rangle = \left(1 - \frac{1}{\|\Pi_M|\psi_0\rangle\|^2}\right)\Pi_M|\psi_0\rangle + (I - \Pi_M)|\psi_0\rangle,$$

so by definition, we can see that $|\psi_0\rangle = |w_{\mathcal{A}}\rangle + |w_{\mathcal{B}}\rangle$, and it remains only to show that $|w_{\mathcal{B}}\rangle \in \mathcal{B}$, by showing that it is orthogonal to $V_{\mathcal{P}}|0\rangle = |\psi_0\rangle$, to establish that this is a negative witness. We thus compute:

$$\begin{aligned}\langle\psi_0|w_{\mathcal{B}}\rangle &= \left(1 - \frac{1}{\|\Pi_M|\psi_0\rangle\|^2}\right)\langle\psi_0|\Pi_M|\psi_0\rangle + \langle\psi_0|(I - \Pi_M)|\psi_0\rangle \\ &= \left(1 - \frac{1}{\|\Pi_M|\psi_0\rangle\|^2}\right)\|\Pi_M|\psi_0\rangle\|^2 + 1 - \|\Pi_M|\psi_0\rangle\|^2 = 0.\end{aligned}$$

Thus, $|w_{\mathcal{A}}\rangle$ and $|w_{\mathcal{B}}\rangle$ form a negative witness, and

$$\||w_{\mathcal{A}}\rangle\|^2 = \frac{\|\Pi_M|\psi_0\rangle\|^2}{\|\Pi_M|\psi_0\rangle\|^4} = \frac{1}{\|\Pi_M V_{\mathcal{P}}|0\rangle\|^2} \leq \frac{1}{\varepsilon} =: \mathcal{C}_-.$$

Thus, with the chosen values of $c_+ = 1$ and $\mathcal{C}_- = \frac{1}{\varepsilon}$, we satisfy the conditions of Theorem 2.2.9. That is, when $\|\Pi_M V_{\mathcal{P}}|0\rangle\|^2 = 0$, we satisfy the positive condition, and when $\|\Pi_M V_{\mathcal{P}}|0\rangle\|^2 \geq \varepsilon$, we satisfy the negative condition, and so we can distinguish these two cases with bounded error in cost

$$O(\mathsf{A} + \sqrt{\mathcal{C}_-}(\mathsf{A} + \mathsf{C})) = O\left(\frac{1}{\sqrt{\varepsilon}}(\mathsf{A} + \mathsf{C})\right).$$

We note that this expression is not exactly a fully worked out complexity, as it depends on the complexities A and C, which might be query complexities, or might count local gates, or local gates plus random access gates. In any particular setting where we might use amplitude amplification (or its decision version), we can plug in the values of A and C, which are complexities in some particular model, to get a complexity in that model.

A more quantum version of the task in Algorithm 6 is where we are looking for a quantum state in some space H_M such that we can implement the reflection $2\Pi_M - I$ (Π_M need not be of the special form $\Pi_M = \sum_{z\in M}|z\rangle\langle z|$). A brute force approach to this task would be to repeat $1/\varepsilon$ times, where $\varepsilon = \|\Pi_M V_{\mathcal{P}}(x)|0\rangle\|^2$: (a) Generate $V_{\mathcal{P}}(x)|0\rangle$; (b) Measure $\{\Pi_M, I - \Pi_M\}$, and output the state if the result is Π_M. However, we can do better with the following generalization of Theorem 2.2.10.

Theorem 2.2.11 (Amplitude Amplification, More General [BHMT02])**.** *Let H be any finite-dimensional inner product space, and $H_M \subset H$ a* marked space*. Suppose we can implement the following quantum subroutines:*

Solution-sample: *Generate a state $V_{\mathcal{P}}(x)|0\rangle$ for some unitary $V_{\mathcal{P}}(x)$ on H, in cost* A

Check: *Implement the reflection $2\Pi_M - I$, where Π_M is the orthogonal projector onto H_M, in cost* C

Let ε be a lower bound on $\|\Pi_M V_{\mathcal{P}}(x)|0\rangle\|^2$. Then there is a quantum algorithm that outputs a δ-approximation to the state proportional to $\Pi_M V_{\mathcal{P}}(x)|0\rangle$ in complexity

$$O\left(\frac{1}{\sqrt{\varepsilon}}(\mathsf{A}+\mathsf{C})\log\frac{1}{\delta}\right).$$

Exercise 2.2.14. *A* block encoding *of a Hermitian matrix A on a space H_1 with norm at most 1, is a unitary U on $H = H_1 \otimes H_2$ such that $A = (I_{H_1} \otimes |0\rangle\langle 0|)U(I_{H_1} \otimes |0\rangle\langle 0|)$. Suppose you have a quantum algorithm that implements $|0\rangle \mapsto |\psi\rangle$ for some state $|\psi\rangle \in H_1$, and a quantum algorithm that implements U. Describe a quantum algorithm for generating a state $|\tilde{\phi}\rangle$ such that*

$$\left\| |\tilde{\phi}\rangle - \frac{A|\psi\rangle}{\|A|\psi\rangle\|} \right\|^2 \leq \frac{1}{4}.$$

Further Directions Just as we can extend Grover's algorithm to approximating the number of ones in the input, x, amplitude amplification can be extended to amplitude *estimation*, where the goal is to output an estimate $\tilde{p}$ of $p = \|\Pi_M V_{\mathcal{P}}|0\rangle\|^2$ such that $|p - \tilde{p}| \leq \delta p$. This can be done in $O(\frac{1}{\delta\sqrt{\varepsilon}})$ calls to $V_{\mathcal{P}}$ and R_M whenever ε is a lower bound on p [BHMT02].

Bibliography

[AGJO+15] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12):123010, 2015. arXiv: 1502.03450 6

[Amb12] Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *Proceedings of the 29th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 636–647, 2012. arXiv: 1010.4458 3

[Bab16] Laszlo Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC)*, pages 684–697, 2016. 18

[BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *Contemporary Mathematics Series*, pages 53–74. AMS, 2002. arXiv: quant-ph/0005055 29, 30, 31

[BJY23] Aleksandrs Belovs, Stacey Jeffery, and Duyal Yolcu. Taming quantum time complexity. arXiv: 2311.15873, 2023. 9

[BS16] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 893–902, 2016. 18

[Chi21] Andrew M. Childs. Lecture notes on quantum algorithms. Available at https://www.cs.umd.edu/~amchilds/qa/, 2021. 13, 18, 19

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 17

[DJ92] David Deutsch and Richard Jozsa. Rapid solutions of problems by quantum computation. 439(1907):553–558, 1992. 1

[EHK04] M. Ettinger, P. Høyer, and M. Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters*, 91(1):43–48, 2004. arXiv: quant-ph/0401083 19

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16):160501, 2008. arXiv: 0708.1879 6

[Hal07] Sean Hallgren. Polynomial-time quantum algorithms for pell's equation and the principal ideal problem. *Journal of the ACM*, (1):653–658, 2007. 18

[HH00] L. Hales and S. Hallgren. An improved quantum fourier transform algorithm and applications. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 515–525, 2000. 5

[HRTS03] Sean Hallgren, Alexander Russell, and Amnon Ta-Shma. The hidden subgroup problem and quantum computation using group representations. *SIAM Jounral on Computing*, 32(4):916–934, 2003. 18

[IMS03] G. Ivanyos, F. Magneiz, and M. Santha. Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem. *International Jounral of Foundations of Computer Science*, 14(5):723–740, 2003. 18

[ISS12] G. Ivanyos, L. Sanselme, and M. Santha. An efficient quantum algorithm for the hidden subgroup problem in nil-2 groups. *Algorithmica*, 62(1–2):480–498, 2012. 18

[Jef22] Stacey Jeffery. Quantum subroutine composition. arXiv: 2209.14146, 2022. 3, 9

[JZ23] Stacey Jeffery and Sebastian Zur. Multidimensional quantum walks and application to k-distinctness. In *Proceedings of the 55th ACM Symposium on the Theory of Computing (STOC)*, pages 1125–1130, 2023. arXiv: 2208.13492 27

[Kit96] Alexei Y. Kitaev. Quantum measurements and the Abelian stabilizer problem. *ECCC*, TR96-003, 1996. arXiv: quant-ph/9511026 15, 19

[Kup05] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005. arXiv: quant-ph/0302112 18

[ME98] Michele Mosca and Artur Ekert. The hidden subgroup problem and eigenvalue estimation on a quantum computer. In *Proceedings of the 1st NASA QCQC conference*, pages 174–188, 1998. 15

[Reg04a] Oded Regev. Quantum computation and lattice problems. *SIAM Journal on Computing*, 33:738–760, 2004. 18

[Reg04b] Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. arXiv: quant-ph/0406151, 2004. 18

[Wat01] John Watrous. Quantum algorithms for solvable groups. In *Proceedings of the 33rd ACM Symposium on the Theory of Computing (STOC)*, pages 60–67, 2001. 18

[dW19] Ronald de Wolf. Quantum computing lecture notes. arXiv: 1907.09415v5, 2019. 1, 3, 4, 5, 10, 17, 18, 19, 21

A.1 Supplementary Material

A.1.1 Group Theory

Recall that a group G is a set of elements with an associated *group operation*, $+$, that has the following properties:

Associativity: For all $g_1, g_2, g_3 \in G$, $g_1 + (g_2 + g_3) = (g_1 + g_2) + g_3$.

Closure under $+$: For all $g_1, g_2 \in G$, $g_1 + g_2 \in G$.

Identity: There is an *identity element* $e \in G$ such that for all $g \in G$, $e + g = g + e = g$.

Inverses: For every $g \in G$, there is an *inverse* $-g \in G$ such that $g + (-g) = e$.

The group is *Abelian* if it also satsifies:

Commutativity: For all $g_1, g_2 \in G$, $g_1 + g_2 = g_2 + g_1$.

An example of an Abelian group is $\mathbb{Z}_N$, the integers modulo N (for some natural number N). An example of a non-Abelian group is S_n, the *symmetric group*, consisting of all permutations of n objects, with composition as the group operation.

A *subgroup* H of G is a subset of G that is also a group with respect to the group operation of G. A (left) *coset* of H is a set

$$g + H = \{g + h : h \in H\}$$

for some $g \in G$. We can also define *right* cosets, which are the same as left cosets in the Abelian case. For any $g \in H$, $g + H = H$, whereas for any $g \notin H$, $g + H$ is not a subgroup, because it does not contain the identity. For all $g \in G$, $|g + H| = |H|$. For any $g, g' \in G$, $g + H = g' + H$ if and only if $g - g' \in H$, and otherwise, $g + H$ and $g' + H$ are disjoint. That is, there are $|G|/|H|$ distinct cosets of H, and they form a partition of G. If $T \subseteq G$ is a set containing exactly one element of each distinct coset of H, called a set of representatives, then every $g \in G$ can be uniquely expressed as $g = h + t$ for some $h \in H$ and $t \in T$.

For a set $S \subset G$, we let

$$\langle S \rangle = \left\{ \sum_{g \in S} a_g g : \{a_g\}_{g \in S} \subset \mathbb{Z} \right\}$$

so S is the smallest subgroup of G containing S. Conversely, for any subgroup H, we say S is a *generating set* for H if $\langle S \rangle = H$. Adding one more element $g \notin \langle S \rangle$ to S at least doubles the size of $\langle S \rangle$, which implies that H always has a generating set of size at most $\log |H|$.

If G and G' are groups, their direct product, $G \times G'$ is the set

$$\{(g, g') : g \in G, g' \in G'\}$$

under the group operation

$$(g_1, g_1') + (g_2, g_2') = (g_1 + g_2, g_1' + g_2').$$

Theorem A.1.1 (Fundamental theorem of finite Abelian groups)**.** *If G is a finite Abelian group, then it is isomoprhic to $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_\ell}$ for some $\ell \in \mathbb{N}$ and $N_1, \ldots, N_\ell \in \mathbb{N}$.*

Theorem A.1.2 (Lagrange's theorem)**.** *If H is a subgroup of G, $|H|$ divides $|G|$.*

For any natural number N, let

$$\mathbb{Z}_N^* = \{g \in \mathbb{Z}_N : \mathrm{GCD}(g, N) = 1\}$$

be the set of elements of $\mathbb{Z}_N$ that are coprime to N, which is a group under multiplication. Its size is denoted $\phi(N) = |\mathbb{Z}_N^*|$. For example, we have $\phi(N) = N - 1$ if and only if N is prime. Otherwise, $\phi(N) < N - 1$.

A.1.2 Binomial distribution

A random variable Z representing the number of successes in k independent trials, where each trial has success probability p is said to have *binomial distribution* $B(k, p)$. Its mean is kp, and we can upper bound the probability of deviating too far from this mean by using a *tail bound*, such as the following:

$$\Pr[Z \leq z] \leq e^{-\frac{2}{k}(pk - z)^2} = e^{-2k\left(p - \frac{z}{k}\right)^2} \tag{A.12}$$

This is a special case of *Hoeffding's inequality*, which gives a tail bound for any random variable that is the sum of independent random variables. Tighter bounds can be obtained from *the Chernoff bound*, but the bound in (A.12) will be sufficient for our purposes.