# Quantum Computing

Summer minicourse
University of Michigan

July 2016

## Disclaimer

The following are rough lecture notes, and are likely rife with typos. See the end for a list of references used, but the notes adhere fairly closely to Ronald de Wolf's excellent set of quantum computing notes.

## Notation

- Elements of vector spaces will be written $|v\rangle$.
- Elements of the dual space will be written $\langle v|$. All our vector spaces will have a (nondegenerate) inner product, so this is well-defined.
- The inner product of two vectors $v, w$ is denoted $\langle v|w\rangle$. Our inner product will be Hermitian, so $\langle v|w\rangle = \overline{\langle w|v\rangle}$. We also write $\langle v|A|w\rangle$ for $\langle v|Aw\rangle$.

## Formalism of QM

To avoid some obnoxious subtleties, all the spaces in question are finite-dimensional. Note this precludes modeling the phenomena of position, momentum, etc. that we're most used to, but works great for quantum computing.

- To an isolated physical system we associate a Hilbert space.
- A point in the Hilbert space represents the "wave function", which describes the physical state of the system.
- Two wavefunctions that differ by a constant lead to the same physical state (corresponds to probablity renormalization and an overall scaling factor).

**Example.** Consider a 2-dimensional Hilbert space with orthonormal basis $\{|0\rangle, |1\rangle\}$, for example modeling the "spin" of an electron along the $z$-axis. An arbitrary state can be written

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

and we'll demand that $|\alpha|^2 + |\beta|^2 = 1$. Such a physical system is called a "qubit", and in some sense this example is the only one we'll discuss.

There are two things we can do to a quantum state: measure something about it, or let it "evolve" (the difference between them is a deep and unanswered philosophical question).

## Measurements

- A measurement $A$ will correspond to a Hermitian operator. (Why Hermitian: real eigenvalues, which is all we measure, and we have orthonormla eigenbases.)
- We will *always* measure an eigenvalue of the Hermitian operator. After we measure, the new state of the system is an eigenvector of the eigenvalue.

- If each eigenvalue has multiplicity 1, it is then clearly determined what state the system is in (since the global phase doesn't matter):
- Say $\{v_1, \ldots, v_n\}$ is an orthonormal basis of eigenfunctions. The probability of measuring a result corresponding to the eigenvector $|v_i\rangle$ is the squared magnitude of the coefficient of $v_i$, i.e., $|\langle v_i|A|\psi\rangle|^2$.

**Example.** Return to our previous example. In the basis $\{|0\rangle, |1\rangle\}$ the operator measuring the "spin" is (up to scalars)

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

Our basis is already an eigenbasis; recall that the state is $\psi = \alpha|0\rangle + \beta|1\rangle$. The chances of measuring 1 are $|\alpha|^2$ and the chances of measuring $-1$ are $|\beta|^2$. (Note that this is why we demanded the normalization condition!) If we measure 1, the state collapses to $|0\rangle$, if we measure $-1$ it collapses to $|1\rangle$.

Let's specialize and say that $\psi = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Now, let's say we have another operator

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

The eigenvalues are still $\pm 1$, and the eigenvectors are $|\pm\rangle := \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. Reexpressing $\psi$ in this basis, we have $\psi = |+\rangle$. Thus, the system is already in an eigenstate of the operator, so we're guaranteed to measure $+1$!

Contrast this with $\psi' = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. In the original basis, this has the same equal measurement probabilities as $\psi$! But $\psi' = |-\rangle$, so measured in the new basis we're guaranteed to get $-1$. This example is in some sense the basis of quantum computing.

## Time evolution

We can act on a quantum state by a unitary transformations (which unitary transformations are "realistic" will be discussed briefly later). Why unitary? We want to preserve the norm of a vector (since physically this must always be 1), so by the polarization identities we need to preserve the inner product, and hence be unitary. Note this implies quantum transformations are reversible!

**Example.** Let

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

(with respect to the basis $\{|0\rangle, |1\rangle\}$). This is clearly unitary. Moreover, note that $U|0\rangle = |+\rangle$ and $U|1\rangle = |-\rangle$. This example will come up many times.

## Multiparticle systems

You can't compute with a single electron. Given two noninteracting systems with associated Hilbert spaces $H_1, H_2$, the total associated Hilbert space is $H = H_1 \otimes H_2$. (This generalizes to arbitrary numbers of particles.) We may as well specialize to $H_1, H_2$ the 2-dimensional vector spaces above. We'll adopt the following notation:

$$|00\rangle := |0\rangle \otimes |0\rangle, \;\; |01\rangle := |0\rangle \otimes |1\rangle, \;\; |10\rangle := |1\rangle \otimes |0\rangle, \;\; |11\rangle := |1\rangle \otimes |1\rangle.$$

Thus an orthonormal basis of $H$ is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. All of the above details still apply, but we should make a note on measurement:

Consider the measurement corresponding to measuring only the first qubit; this corresponds to the hermitian operator

$$\begin{pmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & 1. \end{pmatrix}$$

Say we measure the first qubit the state

$$(1/2)(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) = (1/2)(|00\rangle + |10\rangle - |01\rangle - |11\rangle),$$

and we observe $|1\rangle$, what does the system collapse to? We'll certainly get an eigenstate where the first qubit is in the state $|1\rangle$, but what about the second qubit? We will get the superposition $(1/\sqrt{2})(|10\rangle - |11\rangle)$, i.e., the (renormalized) result of applying the projection to the $|1a\rangle$ subspace.

Note that now "quantum entanglement" can be seen as simply the fact that the elements of the tensor product aren't all pure products. Take

$$\psi := \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

There is no way to write this as

$$\underbrace{(\alpha|0\rangle + \beta|1\rangle)}_{\text{electron 1}} \otimes \underbrace{(\gamma|0\rangle + \delta|1\rangle)}_{\text{electron 2}}.$$

So, what if we measure the first electron? We have a .5 probability of observing the state $|0\rangle$, and moreover if we do measure $|0\rangle$, the state of the system afterwards *must be* $|00\rangle$ (since $|01\rangle$ appears nowhere in the expansion). That is, if we then measure the second electron it must be in state $|0\rangle$. (And likewise for observing $|1\rangle$).

It doesn't matter how far apart these particles are; measuring the system must instanteously collapse the wavefunction. This is the so-called "spooky action at a distance" Einstein, Podolsky, and Rosen were so skeptical of. (Though note this doesn't violate relativity; no information can be transmitted faster than the speed of light.)

## Quantum circuits

The basis for our model of quantum computing will be "gates": a gate is simply a unitary transformation acting on one or more qubits.

### Bitflip

Just flips the two basis states:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

### Phasegate

Alters the phase of $|1\rangle$.

$$R_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$

The special case where $\theta = \pi$ (so the phase on $|1\rangle$ is just negated) is called a phaseflip, denoted $Z$.

### Hadamard

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

### CNOT

This is a two-qubit gate; it performs a bitflip on the second qbit *if* the first qubit is $|1\rangle$, i.e., $|00\rangle \mapsto |00\rangle, |01\rangle \mapsto |01\rangle, |10\rangle \mapsto |11\rangle, |11\rangle \mapsto |10\rangle$. In the basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$:

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

**$n$-bit Hadamard**

Mathematically this is just $H^{\otimes n}$. Physically, this is just $n$ Hadamard gates applied simultaneously to each qubit. First, note that $H^{\otimes n}|0\cdots 0\rangle = \frac{1}{2^{n/2}}(|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle)$, which when expanded is just

$$\frac{1}{2^{n/2}} \sum_{j \in \{0,1\}^n} |j\rangle,$$

the uniform sum over all possible qubit strings.

More generally, if $i \in \{0,1\}^n$, then some symbol-pushing shows that

$$H^{\otimes n}|i\rangle = \frac{1}{2^{n/2}} \sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} |j\rangle.$$

Finally, note that $H^2 = I$, and likewise $(H^{\otimes n})^2 = I$.

## A tiny bit of complexity classes

We'll be very informal in our handling of complexity classes. First, the classical setup: classical circuits are finite directed acyclic graphs using AND, OR, or NOT gates. Some bits are fed into the circuit, and some output bits come out. a circuit family is a sequence of circuits $C_n$. A problem can be solved in polynomial time if there is a circuit family which "solves" the problem for each input size $n$ such that the size of $C_n$ grows polynomially in $n$. The set of polynomial-time problems is denoted $P$.

There is also a classical notion of randomized circuits, which receive some random bits as input. A randomized circuit computes a function if it returns the right value with probability higher than $1/2 + \epsilon$ for some $\epsilon$. We define analogously the family $BPP$ of bounded-error polynomial time problems. It is widely believed that $BPP = P$, and certainly $BPP \supset P$.

Finally, a quantum circuit is a finite directed acyclic graph using certain elementary gates (which gates?). The input are certain qubits, and the output are qubits which we can measure. We then probabilistically obtain an answer. We define $BQP$ as the problems answerable with probability $> 1/2 + \epsilon$ by a family of quantum circuits growing at most polynomially with $n$.

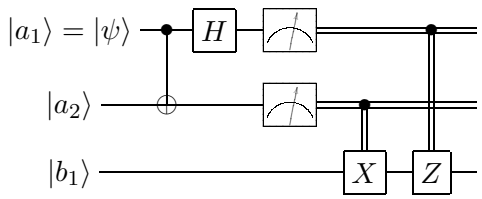
Whether $BQP = BPP$ is unknown. In the upcoming lectures, we'll examine factoring, which is in $BQP$ but is not known to be in $BPP$. If it is not in $BPP$, this provides a counterexample to the "extended Church–Turing hypothesis "Time on all "reasonable" machine models is related by a polynomial.

## Which gates are elementary?

This is partially a physical question of what unitary transformations we can easily effect, so we won't go into too much detail. It is known that the set of all 1-qubit gates and the 2-qubit CNOT gate is universal, but this is an uncountable set! More usefully, the set of CNOT, Hadamard, and the phase gate $R_{\pi/4}$ is "dense", in the sense that any other unitary can be approximated arbitrarily well. Since our algorithms are probabilistic anyways, this is nothing to worry about. Moreover, this approximation is efficient, by the Solovay–Kitaev theorem. (Note: this is actually a theorem about effectively approximating a dense set in $\mathrm{SU}(d)$ using certain generators; see https://www.cs.umd.edu/ amchilds/teaching/w11/l01.pdf). In any case, we'll allow our circuits to have "any" unitary gates, but understand that technically were you to build the circuit physically you'd approximate by CNOTs, Hadamards, and $R_{\pi/4}$.

## Quantum teleportation

The next example is instructive, though perhaps not truly quantum computing. Here is the circuit:




Let's say that Alice has a quantum state $\psi = \alpha|0\rangle + \beta|1\rangle$ she wants to transmit to Bob. They share an entangled pair $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and a classical communication channel (e.g., email). The beginning state is then

$$(\alpha|0\rangle + \beta|1\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle)$$

with Alice with the first two electrons and Bob the third. Alice performs a CNOT gate on her qubits, so the state is

$$\frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle)$$

then a Hadamard transform on the first, so the state is

$$\begin{aligned}\frac{1}{2}(\alpha|000\rangle + \alpha|100\rangle + \alpha|011\rangle + \alpha|111\rangle + \beta|010\rangle - \beta|110\rangle + \beta|001\rangle - \beta|101\rangle) = \frac{1}{2}\big(&|00\rangle(\alpha|0\rangle + \beta|1\rangle)\\ &+|01\rangle(\alpha|1\rangle + \beta|0\rangle)\\ &+|10\rangle(\alpha|0\rangle - \beta|1\rangle)\\ &+|11\rangle(\alpha|1\rangle - \beta|0\rangle)\big)\end{aligned}$$

Remember the last qubit belongs to Bob! Then Alice measures her qubits and transfers the results to Bob. If she gets $|00\rangle$, Bob's qubit collapses to $\alpha|0\rangle + \beta|1\rangle$, the desired state. If instead she measures $|01\rangle$, Bob does a bitflip; if she measures $|10\rangle$ Bob does a phaseflip, and if she measures $|11\rangle$ Bob does both. More simply, if her first qubit is 1, Bob does a phaseflip, and if her second qubit is 1, Bob performs a bitflip.

Note that the state of Alice's original qubit is destroyed. It's easy to see (somewhat informally perhaps) that there is no series of unitary transformations that can "clone" an arbitrary qubit. If there were, it must map $|0\rangle \mapsto |00\rangle$ and $|1\rangle \mapsto |11\rangle$. But then by linearity it would map

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \mapsto \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \neq \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

This is important, because it rules out the use of classical error correction codes that simply duplicate a bit.

## Quantum queries

The last tool we need to discuss before we get down to real algorithms is quantum queries. A query is a "black box" that allows us to access each bit of a bitstring. Let $N = 2^n$, and let $x \in \{0,1\}^N$. The quantum query $O_x$ is a unitary operator on $n+1$ qubits that maps, for each $i \in \{0,1\}^n$ (viewed as a number $\{0,1\}^N$), $|i\rangle|0\rangle \mapsto |i\rangle|x_i\rangle$ (and to preserve unitarity, $|i\rangle|1\rangle \mapsto |i\rangle|x_i \oplus 1\rangle$). That is, we use the first $n$ qubits to address an index in the $2^n$-tuple string $x$.

**Example.** Let $N = 4 = 2^2$, and let $x = 1011$. Then the quantum query $O_x$ maps

$$|00\rangle|0\rangle \mapsto |00\rangle|1\rangle, \quad |01\rangle|0\rangle \mapsto |01\rangle|0\rangle, \quad |10\rangle|0\rangle \mapsto |10\rangle|1\rangle, \quad |11\rangle|0\rangle \mapsto |11\rangle|1\rangle.$$

There's also a related query $O_{x,\pm}$ that maps $|i\rangle \mapsto (-1)^{x_i}|i\rangle$ in the notation above.

**Example.** In our previous example, $O_{x,\pm}$ maps

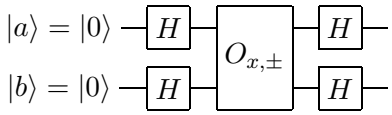
$$|00\rangle \mapsto -|00\rangle, \quad |01\rangle \mapsto |01\rangle, \quad |10\rangle \mapsto -|10\rangle, \quad |11\rangle \mapsto -|11\rangle.$$

In the first few "toy" algorithms we cover, we'll count the number of quantum queries, instead of the number of gates used in the classical model. Note that the speed-ups we'll demonstrate do not hold in the gate-based model! That is, we show quantum speed-ups "relative to the oracle", that is, relative to the existence of the magical query box.

## Deutsch–Jones

Let $N = 2^n$, and say we're given $x \in \{0,1\}^N$ such that either all $x_i$ are the same, or exactly half of them are 0 and half are 1. We want to figure out which of these two possibilities hold (and if they're all the same we don't care which they are). Classically, a deterministic algorithm obviously needs at least $N/2 + 1$ queries. (Of course, there's an obvious probabilistic strategy; more on that later.)

What about quantum mechanically? Consider the following circuit, drawn for the $n = 2$ case:




Recall what happens when we feed $|0^n\rangle$ to $H^{\otimes n}$: we get a uniform sum $(1/\sqrt{2^n}) \sum_{j\in\{0,1\}^n} |j\rangle$. The query $O_{x,\pm}$ then maps each $|j\rangle \mapsto (-1)^{x_j}|j\rangle$. Now, recall what happens when we feed a state $|j\rangle$ through $H^{\otimes n}$, the second set of Hadamards: $|j\rangle \mapsto (1/\sqrt{2^n}) \sum_{k\in\{0,1\}^n} (-1)^{j\cdot k}|k\rangle$.

Now, when we feed the whole state through we get

$$(1/\sqrt{2^n}) \sum_{j\in\{0,1\}^n} (-1)^{x_j}|j\rangle \mapsto (1/2^n) \sum_{j\in\{0,1\}^n} (-1)^{x_j} \sum_{k\in\{0,1\}^n} (-1)^{j\cdot k}|k\rangle.$$

This looks complicated, but consider the coefficient of $|0^n\rangle$ when we expand: for each $j$, $(-1)^{j\cdot|0^n\rangle} = 1$, so the coefficient of $|0^n\rangle$ when we expand is $(1/2^n) \sum_{j\in\{0,1\}^n} (-1)^{x_j}$; by the assumptions on $x$ this is either:

- 0 if half are 1 and half are 0.
- $\pm 1$ if they're all the same.

Thus, let's measure the final state. If all $x_j$ are the same we're *guaranteed* to get $|0^n\rangle$; if not, we're guaranteed not to. Thus, the result tells us the answer *deterministically.*

This algorithm took $2n$ Hadamard gates and 1 quantum query; the corresponding deterministic classical algorithm takes $N/2 + 1 = 2^{n-1} + 1$ classical queries. Of course, for large $N$ a probabilistic algorithm quickly becomes very effective. So here the quantum speedup only concerns the deterministic algorithm.

## Bernstein–Vazirani

The following algorithm uses the *exact* same circuit to solve a different problem. Say $x \in \{0,1\}^N$ such that there is some "hidden" string $a \in \{0,1\}^n$ such that $x_j = j \cdot a \bmod 2$ (where $j$ is taken to be the position in *binary*, hence a vector in $\{0,1\}^n$).

This is perhaps confusing, so an example: Let $n = 2$, so $N = 2^2 = 4$. Say $x = 0110$. Note that if we take $a = 11$, then:

$$\begin{aligned} x_0 &= 00 \cdot 11 = 0, \\ x_1 &= 01 \cdot 11 = 1, \\ x_2 &= 10 \cdot 11 = 1, \\ x_3 &= 11 \cdot 11 = 0, \end{aligned}$$

so $a = 11$ is our desired "hidden" string.

Given such an $x$, how can we find $a$? It turns out the previous algorithm works perfectly. Recall that after the quantum query we have the state

$$\frac{1}{\sqrt{2^n}} \sum_{j \in \{0,1\}^n} (-1)^{x_j} |j\rangle.$$

By assumption $x_j = j \cdot a \bmod 2$, so we can write

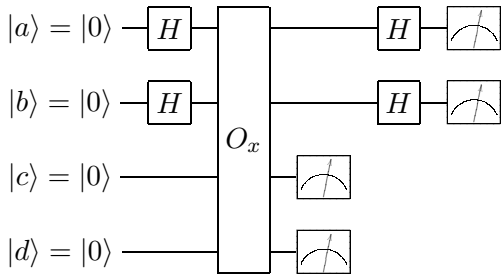
$$\frac{1}{\sqrt{2^n}} \sum_{j \in \{0,1\}^n} (-1)^{x_j} |j\rangle = \frac{1}{\sqrt{2^n}} \sum_{j \in \{0,1\}^n} (-1)^{j \cdot a} |j\rangle.$$

Now, note that the latter is exactly the Hadamard transform of $|a\rangle$! Moreover, the Hadamard transform is its own inverse, so when we apply $H^{\otimes n}$ to this state, we simply get back the pure state $|a\rangle$. Therefore all we need to do is measure the output state, and we're guaranteed to find $a$.

Classically, a query can reveal at most 1 bit of $a$ at a time, so it takes at least $n$ classical queries, whereas we can do it in 1 quantum query!

## Simon's algorithm

Again let $N = 2^n$. Take $X = (x_1, \ldots, x_N)$, with $x_i \in \{0,1\}^n$. That is, $X$ is an $N$-tuple of $n$-bit strings, or, alternatively, a function $\{0,1\}^n \to \{0,1\}^n$. Assume that there is some $s \in \{0,1\}^n$, $s \neq 0^n$, such that $x_i = x_j$ (as bitstrings!) if and only if $i = j \oplus s$, where $\oplus$ is bitwise addition (i.e., addition in the vector space $(\mathbb{Z}/2\mathbb{Z})^n$). That is, $X$ is a 2-to-1 function with "mask" $s$. The goal is to find $s$ with the fewest number of queries. Note that our query will now be a map on $2n$-qubits that maps $|i\rangle|0^n\rangle \mapsto |i\rangle|x_i\rangle$ (since $x_i$ is an $n$-bit string); if you like you can think of this as $n$ single-qubit queries. Below is the circuit diagram, shown for $n = 2$.




The circuit will function as follows: the Hadamards on the first $n$ qubits and the query transform the state as

$$|0^n\rangle|0^n\rangle \xrightarrow{H^{\otimes n} \otimes I} \frac{1}{\sqrt{2^n}} \Big( \sum_{i \in \{0,1\}^n} |i\rangle \Big) |0\rangle \xrightarrow{O_x} \frac{1}{\sqrt{2^n}} \Big( \sum_{i \in \{0,1\}^n} |i\rangle|x_i\rangle \Big).$$

We then measure the second $n$-bit register, yielding some $x_j$. The first $n$-bit register then collapses to a superposition of the two states corresponding to $x_j$, that is, $|j\rangle$ and $|j \oplus s\rangle$:

$$\frac{1}{\sqrt{2^n}} \Big( \sum_{i \in \{0,1\}^n} |i\rangle|x_i\rangle \Big) \xrightarrow{\text{measure}} \frac{1}{\sqrt{2}} (|j\rangle + |j \oplus s\rangle)|x_j\rangle$$

Now, we apply the second round of Hadamards, which yields (by our previous formula)

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{k \in \{0,1\}^n} \big( (-1)^{j \cdot k} + (-1)^{(j \oplus s) \cdot k} \big) |k\rangle.$$

Convince yourselves that $(j \oplus s) \cdot k = (j \cdot k) \oplus (s \cdot k)$ and $(-1)^{a \oplus b} = (-1)^a (-1)^b$, yielding

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{k \in \{0,1\}^n} \big( (-1)^{j \cdot k} + (-1)^{(j \oplus s) \cdot k} \big) |k\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{k \in \{0,1\}^n} (-1)^{j \cdot k} \big( 1 + (-1)^{s \cdot k} \big) |k\rangle.$$

So, when is the coefficient of $|k\rangle$ nonzero? Exactly when $s \cdot k = 0 \bmod 2$, which is a linear equation in the coefficients of $s$. So, we just keep sampling the first $n$ qubits, obtaining different values of $k$, and knowing that each $k$ gives a linear equation satisfied by $s$. $s$ is a nonzero element of the $n$-dimensional vector space $\{0,1\}^n$. Each independent linear equation cuts down the dimension by 1, so we need to find $n-1$ linearly independent conditions on $s$.

How long do we expect this to take? If we've already found $k \leq n-1$ linearly independent vectors, we've described a $k$-dimensional subspace containing $2^k - 1$ nonzero elements. Any nonzero element not in this subspace gives a new condition, and there are $2^{n-1} - 2^k$ such elements we might observe out of $2^{n-1}$ possible elements. Thus the chances of getting a linearly independent condition is

$$\frac{2^{n-1} - 2^k}{2^{n-1}} > 1 - \frac{2^k}{2^{n-1}} \geq \frac{1}{2}.$$

We need to find $n-1$ conditions, so the number of queries we expect this to take is linear in $n$. Thus, we can find $s$ with $O(n)$ many $n$-qubit queries (or $O(n^2)$ many single qubit queries).

### Classical case

With a bit of careful analysis, we can rigorously show the following intuitively obvious result. Classically, the best method is essentially to sample the outputs at random, that is, to check the $x_i$ and hope to find $i, j$ with $x_i = x_j$. How many times do we expect to have to sample? With a bit of thought, you can see that this is essentially the birthday paradox.

Let's say you've examined $T$ of the $x_i$. If any two are the same, we've found $s$, as per above. There are $T(T-1)/2$ possible "match-ups" after taking $T$ samples. Since $s \in \{0,1\}^n$ and $s \neq 0^n$, there are $2^n - 1$ possible choices. Thus, the chances of finding it after $T$ samples is

$$\frac{T(T-1)/2}{2^n - 1} \sim \frac{T^2}{2^{n+1}}.$$

Thus, to find 1 collision we expect it to take roughly $T \sim \sqrt{2^{n+1}}$ queries.

So, the quantum method takes $O(n)$ quantum queries (that is, it's polynomial in $n$), while the classical method is exponential in $n$. Thus, we've demonstrated an exponential speedup (relative to an oracle).

## Shor's algorithm

This algorithm factors a number $N$ (with some high chance of success) in

$$O(\log(N)^2 (\log\log N)^2 \log\log\log N)$$

time, which is only slightly worse than quadratic in the length of the number to be factored. The best known classical (randomized) algorithms (the general number field sieve method) run in

$$2^{\log(N)^\alpha},$$

where $\alpha$ can be shown rigorously to be at most $1/2$ but is believed to be at most $1/3$.

### Reduction to period finding

The setup is as follows: $N$ is a large composite number[1], which we may as well take to be odd, and $x$ is some number $1 < x < N$ which is also coprime to $N$. (Note that Euclid's algorithm allows us to calculate gcds in linear time in the number of input bits, so we're not cheating here.) Consider the sequence

$$x^0 \bmod N, x^1 \bmod N, \ldots.$$

There is of course some smallest value $r$ such that $x^r = 1 \bmod N$ (but since $\mathbb{Z}/N\mathbb{Z}$ is not a group under multiplication $r$ does not need to divide $N$). Let's assume we know $r$.

[1] If $N$ is prime, Shor's algorithm won't tell us anything, but there are fast primality tests, so we may as well assumed we've checked that $N$ is composite.

The following is what we *want* to do:

$$\begin{aligned} x^r &= 1 \bmod N \\ (x^{r/2})^2 &= 1 \bmod N \\ (x^{r/2})^2 - 1 &= 0 \bmod N \\ (x^{r/2}+1)(x^{r/2}-1) &= 0 \bmod N, \end{aligned}$$

and then conclude that $x^{r/2}+1$ and $x^{r/2}-1$ must each share some factor with $N$ which is $< N$, and hence taking $\gcd(x^{r/2}+1, N)$ and $\gcd(x^{r/2}+1, N)$ gives two factors of $N$.

For this to be valid, we need to ensure two conditions: $r$ is even, so $x^{r/2}$ exists, and $x^{r/2} \pm 1$ are not 0 mod $N$, so that $\gcd(x^{r/2} \pm 1, N) \neq N$. We'll show that with probability $\geq 1/2$ a randomly selected $x$ coprime to $N$ will satisfy both these conditions.

First, consider the case where $N = p^e$ (with $p$ odd). Then $G = (\mathbb{Z}/N\mathbb{Z})^\times$ is cyclic of order $p^{e-1}(p-1)$, which is an even number. To see this, note the only elements of $\mathbb{Z}/N\mathbb{Z}$ not coprime to $N$ are $p, 2p, (p^{e-1}p)$. There are $p^{e-1}$ of these, so $|(\mathbb{Z}/N\mathbb{Z})^\times| = p^e - p^{e-1} = p^{e-1}(p-1)$. Write $p^{e-1}(p-1) = 2^d m$.

Let $g$ be a generator of $G$. First say $x = g^l$ with $l$ odd. Then $x^r = g^{lr} = 1 \bmod N$. Then $|G| \mid lr$. Since $l$ is odd, we must have that $2^d \mid r$. Thus at least half of the elements of $G$ have period divisible by $2^d$.

Now let $l$ be even. Note then that $(g^{l/2})^{|G|} = (g^l)^{|G|/2} = 1 \bmod N$, so $r \mid |G|/2 = 2^{d-1}m$, so $2^d \nmid r$. Thus, we see that exactly half the elements of $G$ have order divisible by $2^d$ and half do not.

We'll also need the following calculation: given some $t \leq d$, what are the odds that a random $x \in G$ has order $2^t k$, for any $k$? (That is, what is the order that $2^t$ is the maximal power of 2 dividing $x$?). Again write $x = g^l$. If $x^{2^t k} = 1$, then $g^{2^t kl} = 1$, so $|G| = 2^d m \mid 2^t kl$. Since $k$ is odd, we must have $l = 2^{d-t}c$ for $c$ odd (by minimality of $|x|$). Conversely, any $l$ of this form yields an element with order $2^t k$ for some odd $k$. Thus, we just need to count how many distinct $l$ there are of the form $2^{d-t}c$ for $c$ odd. There are $2^t m$ distinct multiples of $2^{d-t}$ (since after that point they repeat), and half, i.e., $2^{t-1}m$, have even $c$, so there are $(2^t - 2^{t-1})m$ distinct elements of the form $2^t c$ for $c$ odd, and these are the elements with order $2^t k$ for fixed $t$ and any $k$. The chances of an element having order of the form $2^t k$ is then

$$\frac{(2^t - 2^{t-1})m}{2^d m} \leq \frac{(2^d - 2^{d-1})m}{2^d m} \leq 1 - 1/2 = 1/2.$$

Now, take $N = p_1^{e_1} \cdots p_k^{e_k}$. By the Chinese remainder theorem choosing a random element of $\mathbb{Z}/N\mathbb{Z}$ coprime to $N$ is the same as simultaneously and independently choosing an element of each $\mathbb{Z}/p_i^{e_i}\mathbb{Z}$ coprime to $p_i^{e_i}$. Let $r_i$ be the period of $x$ mod $p_i^{e_i}$. It's clear that each $r_i \mid r$. The only way that $r$ can be odd then is if *all* $r_i$ are odd. But as above, the chance of $r_i$ being even is at least $1/2$, so the chance of $r$ being odd is less than $1/2^k \leq 1/4$.

Now, assume that $r$ is even. Finally, note first that $x^{r/2} \neq 1 \bmod N$, since this violates minimality of $r$. Moreover, we claim that $x^{r/2} \neq -1 \bmod N$ with probability at least $1/2$. If $x^{r/2} = -1 \bmod N$, then $x^{r/2} = -1 \bmod p_i^{e_i}$ for each $i$. Then $r_i \nmid (r/2)$ for all $i$. But then we claim that all the $r_i$ must have the same power of 2 in their prime decompositions (assume not: if $r_i = 2^d m$ and $r_j = 2^e n$, with $e > d$, then $2^e m \mid r$, so $2^{e-1}m \mid r/2$, so $r_i = 2^d m \mid 2^{e-1}m \mid r$). Since $r$ is even some $r_i$ is even, so all the $r_j$ are divisible by the same power of 2, but we've shown the odds of $r^j$ being divisible by a particular power of 2 are at most $1/2$ for each $j$. Since we've assumed already that $r$ is even, and hence some $r_i$ is even, the chance of things going wrong at this point is at most $1/4$.

Thus, the chance of either $r$ being odd or $r$ being even and $x^{r/2} = -1 \bmod N$ is $1/4 + 1/4 \leq 1/2$. Therefore if we simply choose $x$ randomly and run the algorithm, checking at the end if it was a "good" $x$, it will only take a few tries (i.e., a number of tries that doesn't influence the complexity class).

## Quantum period finding

### Quantum Fourier transform

First, recall the classical "discrete Fourier transform", a unitary operator which maps $\mathbb{C}^q \to \mathbb{C}^q$:

$$\begin{pmatrix} u_1 \\ \vdots \\ u_q \end{pmatrix} \mapsto \begin{pmatrix} v_1 \\ \vdots \\ v_q \end{pmatrix}, \quad v_j = \frac{1}{\sqrt{q}} \sum_{i=1}^{q} u_i \, e^{2\pi i j/q} := \frac{1}{\sqrt{q}} \sum_{i=1}^{q} u_i \, \omega^{ij},$$

where $\omega$ is a primitive $q$-th root of unity.

The quantum fourier transform is just the discrete fourier transform applied to the vector of amplitudes of basis states. In matrices,

$$F_q = \frac{1}{\sqrt{q}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{q-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(q-1)} \\ 1 & \omega^3 & \omega^6 & \dots & \omega^{3(q-1)} \\ 1 & \omega^{q-1} & \omega^{2(q-1)} & \dots & \omega^{(q-1)^2} \end{pmatrix}.$$

Note that the Hadamard gate is just the $q = 2$ case of this!

One can show (and we will later if time permits) that you can implement the QFT for $n$ qubits (note that this is $q = 2^n$!) in $O(n^2)$ gates (using Hadamards and phase rotation gates), so can use the QFT in polynomial-time algorithms without worrying we're sweeping anything under the rug.

### Modular exponentiation and time complexity

We won't go deeply into the details of how to efficiently compute modular exponentiation on a quantum computer; the algorithm used is classical, and classical circuits can be effectively computed, so this isn't really interesting as a quantum step. The rough idea is to repeatedly square $x, x^2, x^4, x^8$, and then combine these to find $x^i$ for any $i$. We can multiply two $n$-bit numbers mod $N$ via the Scönhage–Strassen algorithm in $O(\log N \log\log N \log\log\log N)$ time, and you can show that breaking the number up into binary representation and multiplying all the squares gives We can do this in $O((\log N)^2 \log\log N \log\log\log N)$ time, and will need to do so approximately $O(\log\log N)$ times. This is the bottleneck of the algorithm, as it runs slower than the quantum fourier transform.

The fourier transform takes $O(n^2) = O(\log(N)^2)$ gates, and we use it $O(\log\log N)$ times (each time we evaluate $f$). This yields

$$O(\log(N)^2 (\log\log N)^2 \log\log\log N)$$

time.

### The circuit

So, let $N$ be the number to be factored. Pick $q = 2^l$ with $N^2 \le q \le 2N^2$ (can always do this: $2M$ is just shifted left one bit, so in between is some number of the form $10\ldots0$.)

Let $n = \lceil \log N \rceil$ (i.e., so $n$ bits is enough to express $f(a) = x^a \bmod N$). Take the following $l + n$-qubit circuit:

We start with the first Fourier transform (which actually behaves identically to the $n$-bit Hadamard on the input state)

$$|0^l\rangle|0^n\rangle \mapsto \frac{1}{\sqrt{2^l}} \sum |a\rangle|0^n\rangle \mapsto \frac{1}{\sqrt{q}} \sum |a\rangle|f(a)\rangle.$$

Now, we measure the second register, obtaining some $f(s)$ (say $s$ is minimal). What elements of the first register are consistent with this second register? Say there happen to be $m$ such distinct elements. Note that $x^a = x^s \bmod N$ exactly when $a - s \bmod N$, so this is the set $M = \{s + jr : 0 \le j < m\}$. Thus after the measurement we have a superposition of $|s\rangle, |s + r\rangle, |s + 2r\rangle, \ldots$:

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr + s\rangle|f(s)\rangle$$

(We'll suppress the second register from here on.) Now, apply the Fourier transform:

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \frac{1}{\sqrt{q}} \sum_{b=0}^{q-1} e^{2\pi i b(jr+s)/q} |b\rangle = \frac{1}{\sqrt{mq}} \sum_{b=0}^{q-1} e^{2\pi i (sb/q)} \left( \sum_{j=0}^{m-1} e^{2\pi i (jrb/q)} \right) |b\rangle.$$

Now, we're going to observe the first $l$ qubits, and get some $|b\rangle$. But what $|b\rangle$ do we expect to see, and what does this tell us about $r$? Consider the coefficient of some arbitrary $|b\rangle$, which has absolute value

$$\left| \frac{1}{\sqrt{mq}} \left( \sum_{j=0}^{m-1} e^{2\pi i (jrb/q)} \right) \right|.$$

If $jrb/q \in \mathbb{Z}$, so $e^{2\pi i (jrb/q)} = 1$, the parenthesized sum is obviously $m$. If the sum is not 1, we just use the formula for the partial sum of a geometric series to see the parenthesized term is

$$\frac{1 - e^{2\pi i (rbm/q)}}{1 - e^{2\pi i (rb/q)}}$$

(Write cases on board here!)

Now, there are two cases to consider.

*Case 1*: The first is easy but unlikely: $r \mid q$. Then $rm = q$ so $m = q/r$. Assume for a second that we're considering a $b$ such that $e^{2\pi i (rb/q)} = 1$. Then $rb/q \in \mathbb{Z}$, so $q/r \mid b$. The squared amplitude is then $(m/\sqrt{mq}) = m/q = 1/r$. There are exactly $r$ such $b$ $(q/r, 2q/r, \ldots, rq/r)$ so actually all of the amplitude is taken up by such $|b\rangle$, and the other states have zero amplitude. Thus only $b$ that are integer multiples of $q/r$ are in the superposition. How do we use this to find $r$?

Let's measure; we'll observe some $b = cq/r$, with $c$ randomly distributed between 0 and $r$. Then $b/q = c/r$. We know $b$ and $q$, but not $c$ and $r$. If $c$ and $r$ are coprime, then we can find $r$ simply by writing $b/q$ in lowest terms. So we just need to try until we get such a $c$ (if we get a nonworking example, we just test the resulting number quickly and see it fails). What are the odds of $c$ being coprime to $r$? To know this, we'll use a number-theoretic estimate on the Euler totient function: $\varphi(r) \in O(1/\log\log r)$ (i.e., $\varphi(r)/\log\log r \to \infty$). Thus we expect to need roughly $\log\log r$ tries before finding such a $c$.

*Case 2*: This is the more likely case, where $r \nmid q$. Now $m \cong \lfloor q/r \rfloor$. We claim that we're likely to see a $|b\rangle$ *close* to an integer multiple of $q/r$ (which is not an integer!). We use the following estimate:

$$|1 - e^{i\theta}| = |e^{i\theta/2}(e^{-i\theta/2} - e^{i\theta/2})| = |2\sin(\theta/2)|,$$

so we can write

$$\frac{1 - e^{2\pi i (rbm/q)}}{1 - e^{2\pi i (rb/q)}} = \frac{|\sin(\pi r b m/q)|}{|\sin(\pi r b/q)|}.$$

We'll bypass some rigorous estimates here, but here's the heuristic: if $b$ is close to a multiple of $q/r$, the argument on the bottom is close to an integer multiple of $\pi$, and hence the denominator is

small. For many such $b$, the top will not be small, since the additional factor of $m$ makes it oscillate "faster".

Precise calculations will show with high probability we will measure a $|b\rangle$ such that

$$\left|\frac{b}{q} - \frac{c}{r}\right| \leq \frac{1}{2q}$$

Again, we know neither $c$ nor $r$.

Note that we chose $q$ such that $N^2 < q \leq 2N^2$, and clearly $r \leq N$. But let $c'/r'$ be a distinct fraction from $c/r$, and assume $r' \leq N$ as well. Then $c/r \neq c'/r'$, so $|cr' - c'r| \geq 1$. But then

$$\left|\frac{c}{r} - \frac{c'}{r'}\right| = \left|\frac{cr' - c'r}{rr'}\right| \geq \frac{1}{N^2} > \frac{1}{q}.$$

Thus $c/r$ is the *only* approximating fraction close enough (i.e., closer than $1/2q$) to $b/q$ with denominator less than $N$.

So we just need a way to obtain such a fraction, but we know $b$ and $q$. We do so via the continued fraction expansion of $b/q$. With good probability (and enough tries) $c$ and $r$ will be coprime, so by writing the approximation in lowest terms we obtain $r$.

## Discrete logarithm problem

There's a related problem, which we'll discuss briefly: the discrete logarithm problem. Let $p$ be a prime, and let $G = (\mathbb{Z}/p\mathbb{Z})^*$. Then $G$ is cyclic; fix some generator $g$. For any $x = g^y$, define $\log_g(x) = y$. The problem is for fixed $p, g$, to calculate $\log_g(x)$ for any $x$. The best classical algorithm is $O(\exp(\log(p)^{1/3}\log(\log(p))^{2/3})$, while we'll demonstrate a quantum algorithm in $O(\log(p)^3)$.

It's essentially a variant of Shor's algorithm for factoring (and was introduced in the same 1993 paper). Define a function $f : \mathbb{Z}/(p-1)\mathbb{Z} \times \mathbb{Z}/(p-1)\mathbb{Z} \to (\mathbb{Z}/p\mathbb{Z})^*$ by $f(a,b) = g^a x^{-b} = g^{a-yb}$. You can verify that $f(a,b) = f(a',b')$ if and only if $(a,b) = (a',b') + \lambda(y,1)$ for some $\lambda \in \mathbb{Z}/(p-1)\mathbb{Z}$. Again, we're going to set up a superposition, apply $f$, then measure some value, then fourier transform to reveal information about $y$.

Here's a sketch; we'll have three registers; in practice each would be $l$ qubits, with $p < 2^l < 2p$ (so they can all represent numbers $\bmod p$). We'll simplify things a bit; first we prepare a uniform superposition over all *pairs* in the first two registers, then apply $f$:

$$\frac{1}{p-1}\sum_{a=0}^{p-1}\sum_{b=0}^{p-1}|a\rangle|b\rangle|0\rangle \mapsto \frac{1}{p-1}\sum_{a=0}^{p-1}\sum_{b=0}^{p-1}|a\rangle|b\rangle|g^{a-yb}\rangle$$

Now, measure the second register, obtaining some $f(a_0, b_0)$. Then the first state collapses to

$$\frac{1}{\sqrt{p-1}}\sum_{k=0}^{p-2}|a_0 + ky\rangle|b_0 + k\rangle|f(a_0,b_0)\rangle$$

Now, again there is some information in the first two registers we want to obtain (the value of $y$), but it's shifted by some $a_0$ and some random $k$; again we use the quantum fourier transform to the first two registers obtain this information. This yields, after some algebra:

$$\frac{1}{\sqrt{p-1}}\sum_{l_1=0}^{p-2} e^{\frac{a_0 l_1 - b_0 y l_2}{p-1}}|l_1\rangle|-yl_2\rangle|f(a_0,b_0)\rangle$$

where $l_2 = -yl_1 \bmod p-1$. Now, we measure the first register, obtaining some $l_1 \in \mathbb{Z}/(p-1)\mathbb{Z}$, and the second, obtaining some number which is "secretly" $-yl_1 \bmod p-1$. If $l_1$ is coprime to $p-1$, then we can (quickly) calculate $l_1^{-1}$, and then obtain $y = -l_2 l_1^{-1}$ by multiplying the result in the second register.

## Hidden subgroup problem

We've had several similar algorithms in a row: Simon, Shor's factoring, and the discrete logarithm. What's going on here?

These are really all examples of hidden subgroup problems.

**Definition.** Let $f : G \to X$ be a function from a group $G$ to some set $X$, and say we have some quantum oracle that can evaluate $f$ on $G$ (where we'll code $G$ as binary strings as before). Let $H \leq G$ be a subgroup of $G$ such that $f$ "hides" $H$, in the sense that $f(g) = f(g')$ if and only if $g, g'$ are in the same coset of $H$. The goal is to find generators for $H$ in $O(\text{poly}(\log(|G|)))$ (i.e., polynomial in the log of the size of $G$).

Note that the naive classical algorithm of just evaluating $f$ on all of $G$ succeeds in $O(\text{poly}\,|G|)$ time, so the goal is to come up with a "feasible" method of finding generators for $H$.

**Example.**
- Simon's algorithm: Here $f : \{0,1\}^n = (\mathbb{Z}/2\mathbb{Z})^n \to (\mathbb{Z}/2\mathbb{Z})^n$, $f(i) = x_i$, with $H = \{0, s\}$ (where $s$ was the "masking" bitstring), and the goal was to find $s$.
- Shor's algorithm: Here $G = \mathbb{Z}/q\mathbb{Z}$, $X = \mathbb{Z}$ or $\mathbb{Z}/N\mathbb{Z}$, with $f(a) = x^a \bmod N$, and $H = \langle r \rangle$.
- Discrete log: $G = \mathbb{Z}/(p-1)\mathbb{Z} \times \mathbb{Z}/(p-1)\mathbb{Z}$, $X = (\mathbb{Z}/p\mathbb{Z})^*$, $f(a,b) = g^a x^{-b}$, and $H = \langle (y, 1) \rangle$.

So in general what can be said about this kind of problem? First, note that the subgroup $H$ is important, but the function $f$ not so much. First, let's do an example showing the power of this formulation. Recall the graph isomorphism problem, which is to decide if two finite graphs are isomorphic, and is known to be $NP$ and believed to not be in $P$. (It's very similar to integer factorization in this way). If we could solve the hidden subgroup problem for a certain subgroup of $S_n$,

So, let $A, B$ be the two graphs, connected of size $n$, and let $C = A \sqcup B$. Let $K = \text{Aut}(C)$. Then certainly $K < S_{2n}$. Let $H = S_n \times S_n$, and let $\sigma = (1\ n+1)(2\ n+2)\dots(n\ 2n)$ (i.e., just swapping the labels of the two graphs). Note that certainly $K \leq H \cup \sigma H$. If $A \ncong B$ then $K \subset H$ entirely (since we can't swap between the two disjoint components), while if $A \cong B$ then half of $K$ is in $H$ and half is in $\sigma H$. Thus, if we observe all the generators of $K$ are in $H$, for example, we'd have solved the graph isomorphism problem (it's easy to see if a generator is in $H$ vs. $\sigma H$; just check if it sends the first vertex to the first $n$ or last $n$ vertices!)

We can place this in the formalism of the hidden subgroup problem with the function $f : G \to M_{2n \times 2n}(\{0,1\})$ sending $g \mapsto g.M_C$, where $M_C$ is the adjacency matrix of $C$ and $g.M_C$ is the adjacency matrix of $g.C$. Then clearly $f(g) = f(g')$ if and only if $g, g'$ are in the same coset of the automorphism group of $C$.

However, there have been demonstrated obstacles to this working: Moore, Russell, and Schulman (2005) showed that the hidden subgroup problem for arbitrary subgroups of $S_n$ cannot be solved via "strong Fourier sampling", which is the form of algorithm we've used so far (i.e., preparing a superposition of inputes, applying the map $f$, taking a Fourier transform, and then measuring). Thus, we'd need a new kind of algorithm, which depends on further entanglement between the different prepared states.

Similarly, a solution to the hidden subgroup problem for $D_n$ (the dihedral grup) would solve the "shortest vector problem", where we're given a basis for a vector space and a norm, as well as a lattice, and we're to find the shortest nonzero vector in the lattice.

Now, we sketch the beginning of an algorithm; note the strong resemblance to Simon's algorithm and Shor's algorithm. We first prepare a uniform superposition

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

then apply the function $f$ to get

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

We then measure the second register, obtaining some $|f(g_0)\rangle$ and collapsing the first to a superposition over some coset of $H$:

$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |g_0 + h\rangle |f(g_0)\rangle,$$

Now, the states present have some information we want (they're the sum over a coset of $H$), but the offset prevents us from analyzing $H$ on its own. So we want to get rid of it; we'll sketch how to solve this in the *abelian* case. It turns out the crucial ingredient making this case tractable is the representation theory of abelian groups:

Recall that an abelian group $G$ has $|G|$ many irreducible representations, and each is 1-dimensional. (Quick recap: a representation of $G$ is just a homomorphism $G$ to the automorphisms of some vector space, i.e., a group action of $G$ on a vector space.) Let $\widehat{G} = \{\chi_1, \ldots, \chi_{|G|}\}$ be the set of irreducible *characters* of $G$ (the character of a 1-dimensional representation is just a map $G \to S^1$).

Now, we'll define an analogue of the quantum fourier transform: Define new vectors by

$$|\chi_I\rangle = \frac{1}{|G|} \sum_{g \in G} \chi_I(g) |g\rangle.$$

You can check that this is actually again a basis for the vector space.

Define the Fourier transform $F_G$ for $G$ as the (unitary!) map $F_G(|\chi_g\rangle) = g$ (note this depends on fixing some identification of $G$ and $\widehat{G}$).

As an example, note that for $\mathbb{Z}/N\mathbb{Z}$ the irreducible representation's values are given by

$$\chi_k(j) = e^{2\pi i j k/N},$$

and

$$|\chi_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i j k/N} |j\rangle,$$

which looks exactly like the states in Shor's algorithm.

For $(\mathbb{Z}/2\mathbb{Z})^n$, the corresponding quantum fourier transform is just the $n$-bit Hadamard!

Now, we apply $F_G$ to this state. First of all, it turns out the amplitudes of the states in

$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |g_0 + h\rangle$$

and

$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |h\rangle$$

are the same, up to phase factors!

Thus if we measure at this point, before introducing further entanglement, all the observation probabilities are equal. This, of course, is exactly the point of the Fourier transform, in that it removes the "random" offset by $g_0$; it turns translation into multiplication by a complex phase, and then quantum mechanics ensures that this doesn't affect measurement probabilities.

But what are the probabilities then? We won't show the details, but it turns out

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

But then the orthogonality of irreducible representations of $H$ implies that

$$\left(\sum_{h \in H} \chi_g(h)\right) = 0$$

unless $\chi_g|_H$ is the trivial representation. Thus, if we observe some $|g\rangle$, we know that $\chi_g|_H$ is the trivial representation. If $r$ generates $K$ alone, then this is equivalent to $\chi_g(r) = 1$. Thus, by

observing the $g$ that we observe, and checking for which $r \in G$ we have $\chi_g(r) = 1$, we can gather evidence about elements $r \in K$.

This is the end of the quantum computational part of the algorithm; the rest of the algorithm depends on using the information about the particular group in question to turn this information into information about the generators of $H$, and this is specific to the group in question (e.g., the coprimality step in Shor's algorithm).

## Grover's algorithm

Grover's algorithm searches an unordered list of $N = 2^n$ items and returns an item satisfying some criterion. We model this as looking at some $x \in \{0,1\}^N$, and we'll assume we have a quantum oracle for $x$. We want to find some $x_i$ such that $x_i = 1$. Clearly, any classical algorithm must take $O(N)$ queries; in contrast, we can do so with a quantum oracle in $O(\sqrt{N})$ queries (with some high probability of success, which does not grow with $N$). Say that $t$ is the number of $x_i$ equal to 1; we'll assume $t << N$.

So, each query takes $2n$ Hadamards, a query, and $O(n)$ from the phase shift, hence overall $O(n)$, and we're performing $O(\sqrt{N}) = O(2^{n/2})$ iterates, so the overall runtime is $O(2^{n/2}n) = O(\sqrt{N}\log N)$.

The algorithm works as follows: We'll have an $n$-qubit circuit. recall the signed quantum query $O_{x,\pm}$ which maps $|i\rangle \mapsto (-1)^{x_i}|i\rangle$. Define a unitary map

$$R = \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & \ddots \end{pmatrix}$$

That is, $R$ puts a phase of $-1$ on everything basis state except $|0^n\rangle$ (this can be efficiently constructed from elementary gates). Define the Grover iterate as the unitary map

$$F = H^{\otimes n} R H^{\otimes n} O_{x,\pm}$$

That is, we query the oracle, apply the $n$-bit Hadamard, then the map $R$, then apply the Hadamard again.

Our algorithm will first prepare a uniform superposition over all $n$-bit strings (again, via the $n$-bit Hadamard), and then apply the Grover iterate some number of times, and we'll then measure the result.

Now, we'll analyze what's going on; we'll see that by the repeated iterates we're "concentrating" amplitude in the states $|i\rangle$ for which $x_i = 1$.

Define

$$|G\rangle = \frac{1}{\sqrt{t}} \sum_{i:x_i=1} |i\rangle$$

and

$$|B\rangle = \frac{1}{\sqrt{t-n}} \sum_{i:x_i=0} |i\rangle.$$

Thus the goal is to concentrate amplitude in $|G\rangle$. Now, let

$$|U\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle = \sqrt{\frac{t}{N}}|G\rangle + \sqrt{\frac{N-t}{N}}|B\rangle.$$

Let $\theta = \arcsin\sqrt{t/N}$. Then we can write

$$\sqrt{\frac{t}{N}}|G\rangle + \sqrt{\frac{N-t}{N}}|B\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle.$$

Now, consider the two-dimensional space spanned by $|B\rangle$ and $|G\rangle$, which we think of as orthogonal vectors. First, consider $O_{x,\pm}$: this fixes all the states for which $x_i = 0$ and negates the rest, so it maps $G \mapsto -G$ and $B \mapsto B$; thus $O_{x,\pm}$ is a reflection across $B$.

Now consider $H^{\otimes n} R H^{\otimes n}$; we can write $R = 2P - I$, where $P$ is the projection to $|0^n\rangle$ (in physics we'd write $P = |0^n\rangle\langle 0^n|$); then

$$H^{\otimes n} R H^{\otimes n} = 2H^{\otimes n} P H^{\otimes n} - I.$$

Now, consider $H^{\otimes n} P H^{\otimes n}|i\rangle$; the coefficient of $|0^n\rangle$ is always $1/\sqrt{N}$ in $H^{\otimes n}|i\rangle$, so this is just $(1/\sqrt{N})H^{\otimes n}|0^n\rangle = (1/\sqrt{N})U$.

Thus, we have that $(H^{\otimes n} P H^{\otimes n})U = U$ and if $V \in \operatorname{span} U^{\perp}$ then $H^{\otimes n} P H^{\otimes U} V = 0$ (since we'll have an even number of negative signs). Thus $H^{\otimes n} P H^{\otimes n}$ is projection to $\operatorname{span} U$, thus $2H^{\otimes n} P H^{\otimes n} - I$ is reflection across $U$.

Thus the Grover iterate is actually two reflections: the first across $B$ and the second across $U$.

Let's analyze what the Grover iterate does to

$$\sin(\theta)|G\rangle + \cos(\theta)|B\rangle.$$

(Draw picture)

Thus each Grover iterate increases the angle by $2\theta$. Thus after $k$ iterations the state will be

$$\sin\big((2k+1)\theta\big)|G\rangle + \cos\big((2k+1)\theta\big)|B\rangle.$$

Thus the probability of measuring a good state after $k$ iterations is

$$P_k = \sin\big((2k+1)\theta\big)^2$$

Thus, we want to pick some $k$ close to $K = \pi/(4\theta) - 1/2$ to maximize this amplitude. We can of course choose $|k - K| \leq 1/2$. We have

$$\begin{aligned} 1 - P_k &= \cos((2k+1)\theta)^2 = \cos((2K+1)\theta + 2(k-K)\theta)^2 \\ &= \cos((2K+1)\theta + 2(k-K)\theta)^2 = \cos(\pi/2 + 2(k-K)\theta)^2 = \sin(2(k-K)\theta)^2 \leq \sin(\theta)^2 = t/N \end{aligned}$$

(since we picked $K$ such that $(2K+1)\theta = \pi/2$)

Note that $k \leq \pi/(4\theta) = \pi/4\sqrt{N/t} \in O(\sqrt{N})$. Note also some slightly odd behavior; as would be intuitively true, $k$ decreases with $t$ (the more correct solutions, the quicker the algorithm!). However, using the above analysis, the error probability also increases with $t$!

Do note that we picked $k$ using knowledge of $\theta$, which depends in turn on knowing how many solutions we expect; if we increase $k$ too much, then the success probability goes down again! You can, however, modify the algorithm to "search" over $k$, obtaining a high probability of finding a solution without affecting the final run-speed too much.

### Amplitude amplification

Note that we can actually use this exact same idea in a more general setting. Assume we have some function $\chi : \mathbb{Z} \to \{0, 1\}$. We want to find some $z \in \mathbb{Z}$ with $\chi(z) = 1$. Assume we have a quantum oracle $|x\rangle \mapsto (-1)^{f(x)}|x\rangle$. Say also we have a algorithm $A$ that returns a quantum state with a probability $p$ of finding a solution when applied to the starting state $|0^n\rangle$. Classically, we'd need

to run $A$ $1/p$ times to find a solution. However, we can use amplitude amplification to run it only $O(1/\sqrt{p})$ times.

The analysis is essentially the same: let $U = A|0\rangle$. Then repeatedly apply $O_\chi$ (reflect through $|B\rangle$) and $ARA$ (reflect through $U$) $O(1/\sqrt{N})$ times, and then measure; the above analysis goes through with $\theta = \arcsin(\sqrt{p})$.

The analogue here is the Hadamard transform is a "search algorithm" with probability $t/N$ for the unordered search, since it returns a uniform superposition of all states.

References


[1] Aaronson, S., "Quantum computing since Democritus", Cambridge University Press, 2013

[2] Bacon, D., *CSE 599d: Quantum computing, Shor's algorithm*, 2006, https://courses.cs.washington.edu/courses/cse599d/06wi/lecturenotes11.pdf

[3] Joszsa, R., *Quantum factoring, discrete logarithms and the hidden subgroup problem*, 2000, arXiv:quant-ph/0012084

[4] Lomont, C., *The Hidden Subgroup Problem — Review and Open Problems*, 2004, arXiv:quant-ph/0411037

[5] Nielsen, M. and Chuang, I., "Quantum Computation and Quantum Information", Cambridge University Press, 2001

[6] Shor, P., *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM Journal on Computing 26, no. 5, 1997, p. 1484-1509.

[7] de Wolf, R., *Quantum computing: Lecture notes*, 2011, http://homepages.cwi.nl/∼rdewolf/qcnotes.pdf