Title: QI Basics

Date: Jul 19, 2016 09:00 AM

URL: http://pirsa.org/16070012

Abstract:

Quantum Computing Basics

John Watrous
Institute for Quantum Computing
University of Waterloo

Part 1

Integer factoring, the quantum circuit model,
and simulating classical computations.

Integer factoring

The fundamental theorem of arithmetic states that every integer $N \geqslant 2$ can be expressed as a product of prime numbers in a unique way (up to the ordering of the prime numbers).

For example:

$$15 = 3 \cdot 5$$

$$198 = 2 \cdot 3^2 \cdot 11$$

$$\begin{array}{c} 12301866845301177551304949583849627207728535 \\ 69595334792197322452151726400507263657518745 \\ 20219978646938995647494277406384592519255732 \\ 63034537315482685079170261221429134616704292 \\ 14311602221240479274737794080665351419597459 \\ 856902143413 \end{array} = \begin{array}{c} 33478071698956898786044169848212690817704794 \\ 98371376856891243138898288379387800228761471 \\ 165253174308773781446799948 9 \\ \cdot \\ 36746043666799590428244633799627952632279158 \\ 16434308764267603228381573966651127923337341 \\ 71433968102700927987363089 17 \end{array}$$

The **integer factoring problem** is to output the expression of a given positive integer (represented in binary notation) as a product of prime numbers.

The importance of computational efficiency

If we are interested in solving a particular computational problem, it is obviously not enough to have an algorithm for solving that problem. We need an algorithm to be **efficient** if it is to be useful.

For example, here is a high-level description of an algorithm for outputting the smallest prime factor of a given integer $N \geqslant 2$:

For k from 2 to N do

If k divides N then output k and stop

The running time for this algorithm is $O(N(\log N)^2)$, assuming a straightforward way of testing if k divides N. This running time is **exponential** in the input length, which is roughly $\log N$.

For a 1000-bit number equal to the product of two 500-bit primes, the algorithm would take many times the age of the universe on the fastest (classical) computer on this planet.

Better classical algorithms for factoring

The algorithm for finding prime factors from the previous slide is inefficient, but this does not imply factoring cannot be solved efficiently.

The fastest known classical algorithm is the **number field sieve**, which (with some heuristic assumptions) runs in time

$$c^{(\log N)^{1/3}(\log\log N)^{2/3}}$$

for some constant $c > 0$.

No **polynomial-time** factoring algorithm is known—such an algorithm would run in time $O((\log N)^c)$ for some fixed constant c.

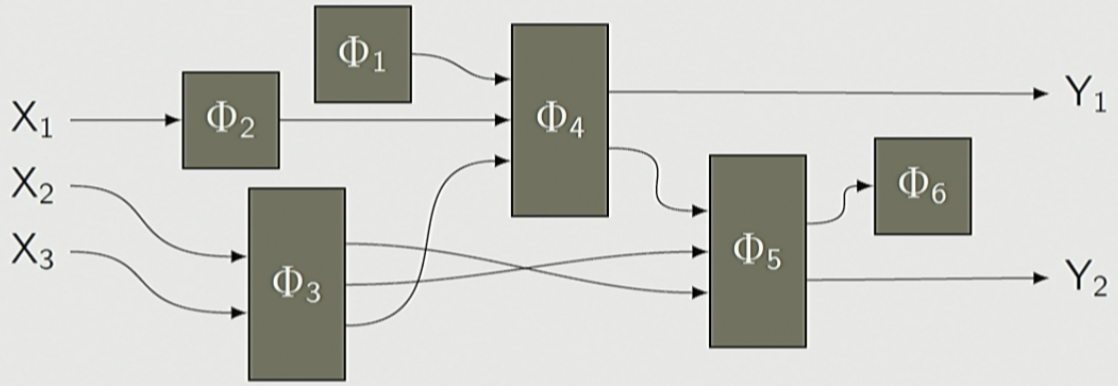
On the other hand, it has not been proved that no polynomial-time algorithm for factoring with a classical computer exists. Proving **lower bounds** on the running time of algorithms is an extremely difficult challenge within theoretical computer science.

Formalizing the notion of efficiency

There are different ways to formalize the number of steps (or elementary operations) a given algorithm requires.

We'll focus on quantum algorithms in the **quantum circuit model**.

In its most general formulation, a quantum circuit is any acyclic network of constant-size quantum operations (or channels) on qubits, like this:



The total number of operations (or **gates**) represents the running time of the algorithm implemented by the circuit.

Efficient quantum circuit algorithms

Suppose that a computational problem of interest is represented by a collection of functions $\{f_1, f_2, f_3, \ldots\}$, where

$$f_n : \{0,1\}^n \to \{0,1\}^m$$

We say that we have an **efficient quantum algorithm** for this problem if there exists a set of quantum circuits $\{Q_1, Q_2, Q_3, \ldots\}$ with these two properties:

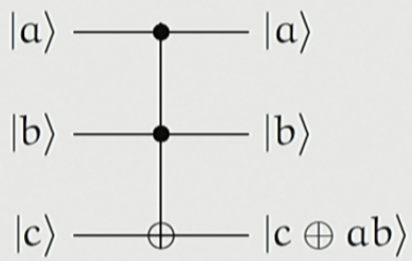
1. For each $x \in \{0,1\}^n$, if we apply Q_n to $|x\rangle$ and then measure with respect to the standard basis, we obtain $f_n(x)$ with high probability.
2. There is a constant c such that $\text{size}(Q_n) = O(n^c)$. In other words, Q_n has size polynomial in n.

Technically speaking, we also need uniformity constraints on the set $\{Q_1, Q_2, Q_3, \ldots\}$. That is, there should be an efficient classical algorithm that computes a description of Q_n for a given input length n.

Unitary circuits

When discussing quantum algorithms, we commonly restrict our attention to **unitary** gates and circuits. A typical **universal gate set**:

Toffoli gate



Hadamard gate

$$H|a\rangle = \frac{|0\rangle + (-1)^a |1\rangle}{\sqrt{2}}$$

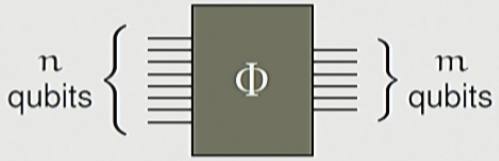
Phase gate

(Also called a $\pi/4$ gate.)

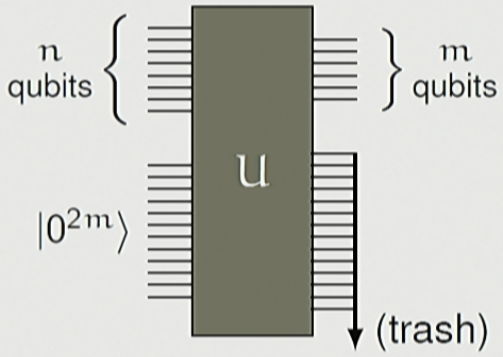
(A universal gate set is one that allows arbitrarily good approximations to arbitrary gates.)

Unitary operations can simulate arbitrary operations

It is always possible to simulate a general quantum operation by a unitary operation. Take any operation of the following form:



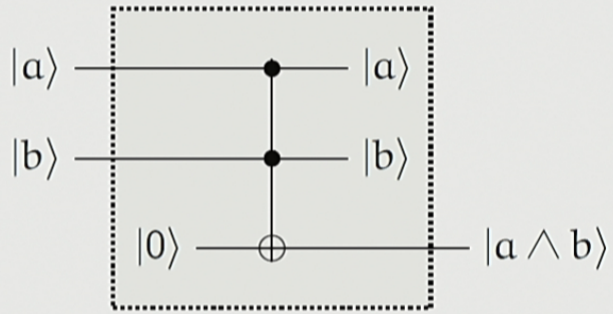
There is a unitary operation U that induces this operation as follows:



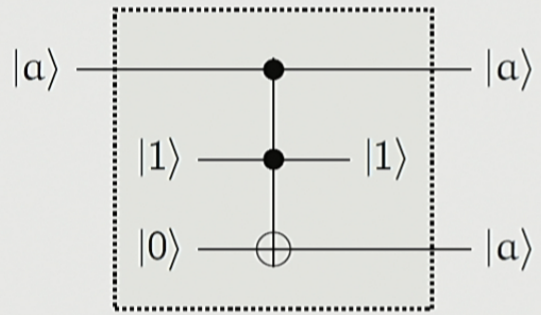
Simulating classical circuits

It is not difficult to show that classical Boolean circuits can be simulated by quantum circuits. It is enough to consider the following gates:

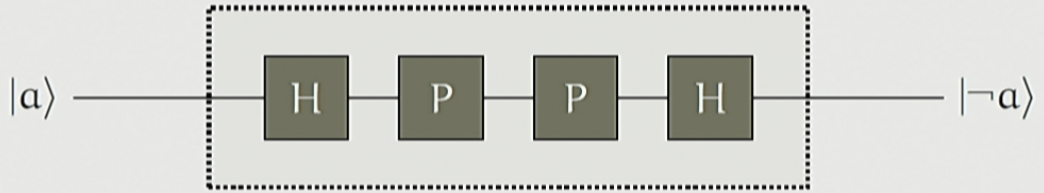
AND gates



FANOUT gates



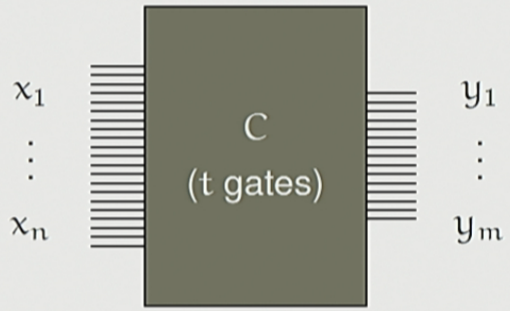
NOT gates



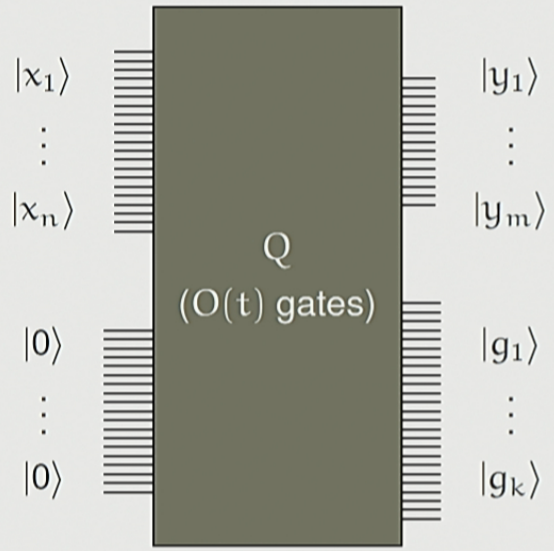
Simulating classical circuits

One obtains a simulation of a classical Boolean circuit as follows:

Original circuit

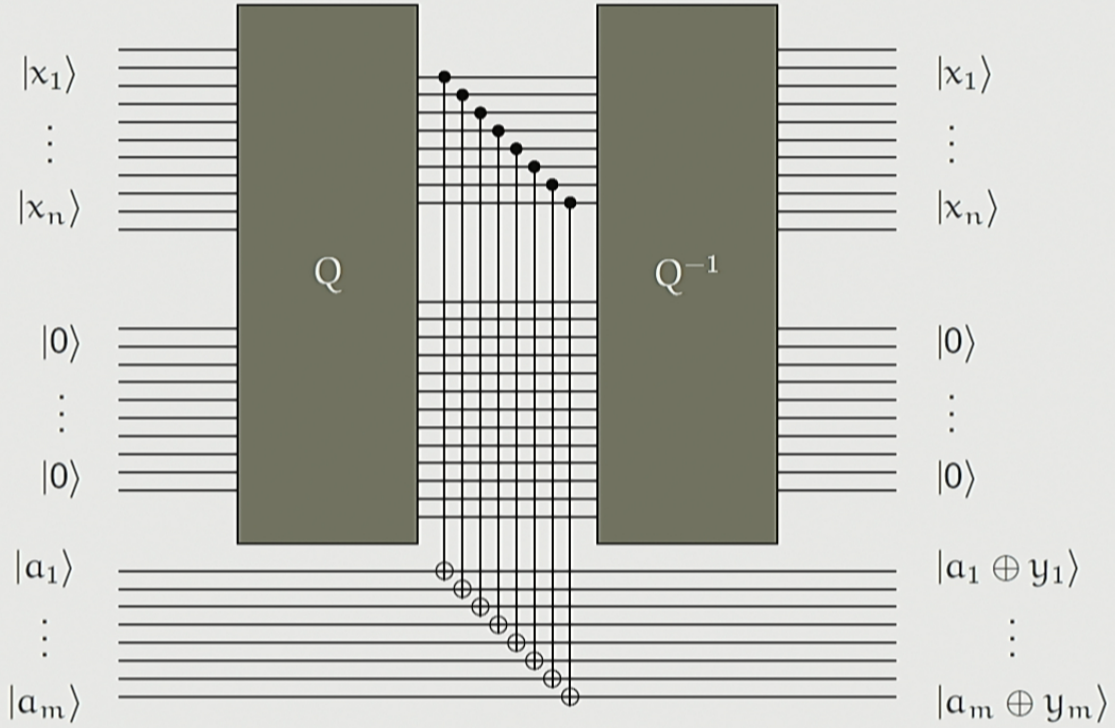


Unitary circuit



Simulating classical circuits

Given a unitary circuit Q that deterministically computes a function $y_1 \cdots y_m = f(x_1 \cdots x_n)$ with garbage, one can obtain a **garbage-free** implementation as follows:



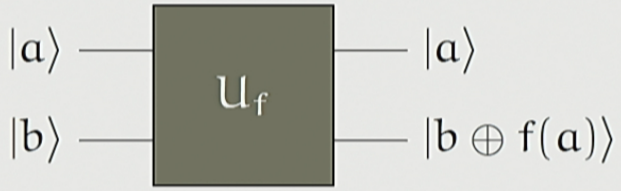
Part 2

Deutsch's algorithm; eigenvalue estimation

Deutsch's problem

A simple computational problem, called **Deutsch's problem**, illustrates an advantage of quantum over classical algorithms.

We are given a function $f : \{0, 1\} \to \{0, 1\}$, in the form of a black box. Specifically, we are given access to its unitary implementation:



The goal is to determine to which of the following categories f belongs:

Constant

a	$f(a)$
0	0
1	0

a	$f(a)$
0	1
1	1

Balanced

a	$f(a)$
0	0
1	1

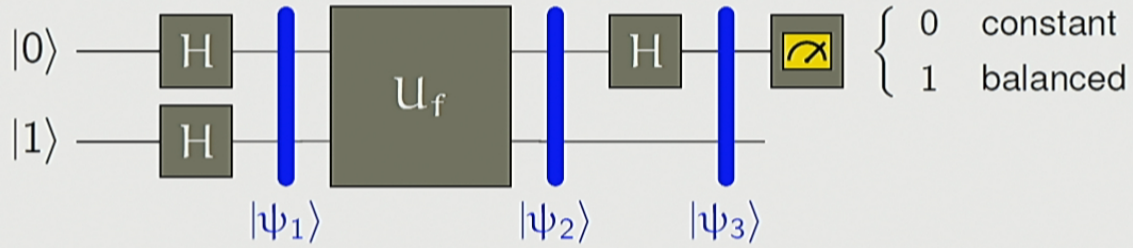
a	$f(a)$
0	1
1	0

Query complexity of Deutsch's problem

How many uses of the black box are required to solve Deutsch's problem?

Classical algorithm: 2 queries are necessary (and sufficient).

Quantum algorithm: a single query suffices. Here is the algorithm:



States of the circuit:

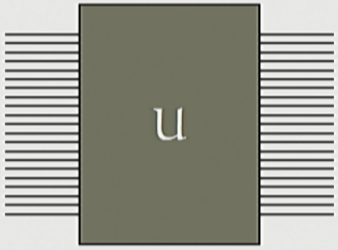
$$|\psi_1\rangle = \frac{1}{2}\big(|0\rangle + |1\rangle\big)\big(|0\rangle - |1\rangle\big)$$

$$|\psi_2\rangle = \frac{1}{2}\big((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\big)\big(|0\rangle - |1\rangle\big) \quad \leftarrow \text{“phase kick-back”}$$

$$|\psi_3\rangle = \left(\frac{(-1)^{f(0)} + (-1)^{f(1)}}{2}|0\rangle + \frac{(-1)^{f(0)} - (-1)^{f(1)}}{2}|1\rangle\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Spectrum of a unitary operation

Suppose U is a unitary operation on n qubits.



As a linear operator, U acts on a space of dimension $N = 2^n$.

Because U is unitary, we know from linear algebra that it has N **orthogonal eigenvectors**

$$|\psi_1\rangle\,,\ |\psi_2\rangle\,,\ \ldots\,,\ |\psi_N\rangle$$

with corresponding **eigenvalues**

$$\lambda_1 = e^{2\pi i\theta_1},\ \lambda_2 = e^{2\pi i\theta_2},\ \ldots,\ \lambda_N = e^{2\pi i\theta_N}.$$

Eigenvalue estimation problem

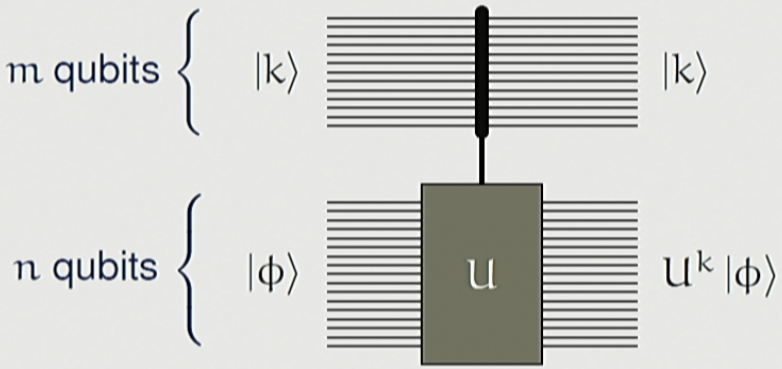
The **eigenvalue (or phase) estimation problem** is as follows:

Input: a quantum circuit implementing U, and an eigenvector $|\psi\rangle$ of U (as a quantum state).

Goal: compute (or approximate) $\theta \in [0, 1)$ such that

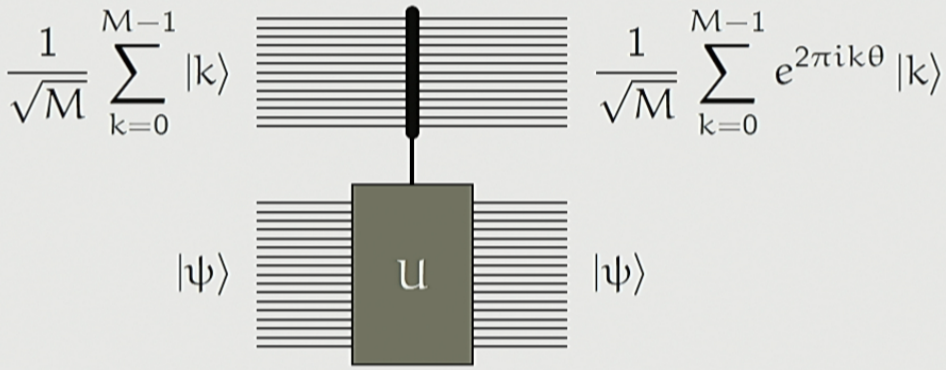
$$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle .$$

In general, we only know how to efficiently solve this problem with high accuracy if we can efficiently implement a $\Lambda_m(U)$ transformation:



Action of controlled unitary on eigenvectors

Consider the action of a $\Lambda_m(U)$ transformation when the lower register is set as an eigenvector of U:



In summary, we obtain this transformation:

$$\Lambda_m(U):\left(\frac{1}{\sqrt{M}}\sum_{k=0}^{M-1}|k\rangle\right)\otimes|\psi\rangle\mapsto\left(\frac{1}{\sqrt{M}}\sum_{k=0}^{M-1}e^{2\pi ik\theta}|k\rangle\right)\otimes|\psi\rangle$$

for $M=2^m$.

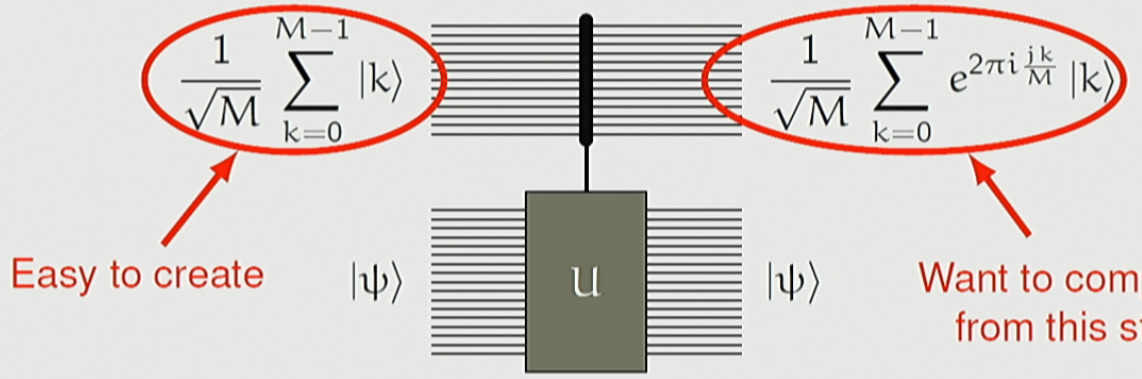
An easy special case

Let us consider the special case in which

$$\theta = \frac{j}{2^m}$$

for some choice of $j \in \{0, \ldots, M-1\}$. (In general, we cannot assume θ takes this form.)

From the previous slide:



An easy special case

Our goal is to compute j from the state

$$|\phi_j\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} e^{2\pi i \frac{jk}{M}} |k\rangle ,$$

for each $j \in \{0, \ldots, M-1\}$.

For this to be possible, the set of states $\left\{|\phi_0\rangle , \ldots , |\phi_{M-1}\rangle\right\}$ must be **perfectly distinguishable** (i.e., form an orthonormal set). It can be shown that this is indeed an orthonormal set, so it is possible in principle to accomplish our goal...

We require a unitary transformation V such that

$$V |\phi_j\rangle = |j\rangle ,$$

so that we may simply apply V to $|\phi_j\rangle$ and measure in the standard basis to determine j.

The required transformation

The requirement that V satisfies $V|\phi_j\rangle = |j\rangle$, or equivalently

$$V^\dagger |j\rangle = |\phi_j\rangle\,,$$

for all $j \in \{0, \ldots, M-1\}$, is enough to uniquely determine V. As a matrix, we have

$$V^\dagger = \frac{1}{\sqrt{M}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{M-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(M-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{M-1} & \omega^{2(M-1)} & \cdots & \omega^{(M-1)^2} \end{pmatrix}$$

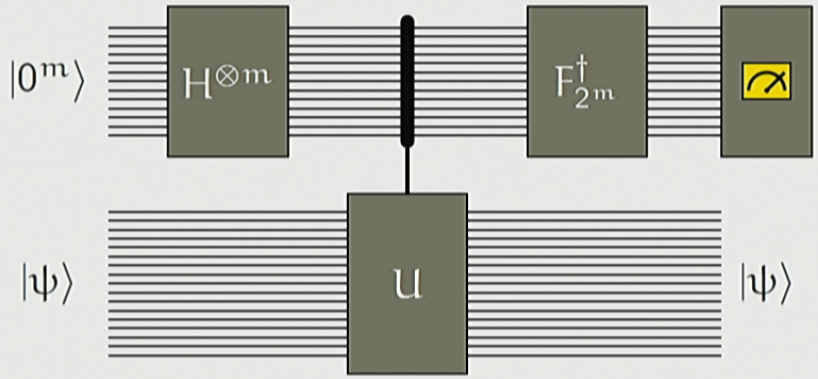
for $\omega = e^{2\pi i/M}$.

This is a familiar matrix to many. . . it represents the mapping known as the **discrete Fourier transform**.

The eigenvalue estimation algorithm

We refer to the unitary operation V from the previous slide, which applies the discrete Fourier transform to an M dimensional state vector, as the **quantum Fourier transform**, and will denote it F_M.

The final algorithm looks like this:



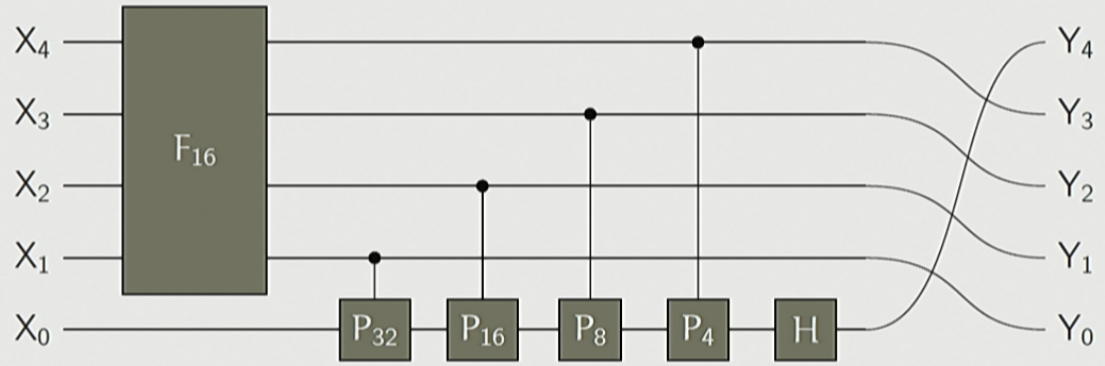
In the special case that $\theta = \frac{j}{2^m}$ for $j \in \{0, \ldots, 2^m - 1\}$, the measurement (in the standard basis) results in outcome j (in binary) with certainty.

Two remaining issues

There are two remaining issues to consider:

1. We need to know how to implement the quantum Fourier transform F_{2^m} efficiently.
2. We do not yet know what happens in the case that θ does not take the form $j/2^m$ for $j \in \{0, \ldots, 2^m - 1\}$.

The first issue is addressed by a recursive construction requiring $O(m^2)$ gates in total. For example, F_{32} is implemented like this:



The phase gates are defined as $P_K : |a\rangle \mapsto e^{2\pi i a/K} |a\rangle$.

General eigenvalues

The last issue remaining is that θ may not take the form $j/2^m$ for $j \in \{0, \ldots, 2^m - 1\}$. We do not need to change the algorithm to handle this case...

For a general value of $\theta \in [0, 1)$, the final state of the top m qubits (before the measurement) is

$$\frac{1}{M} \sum_{j=0}^{M-1} \left(\sum_{k=0}^{M-1} e^{2\pi i k(\theta - j/M)} \right) |j\rangle$$

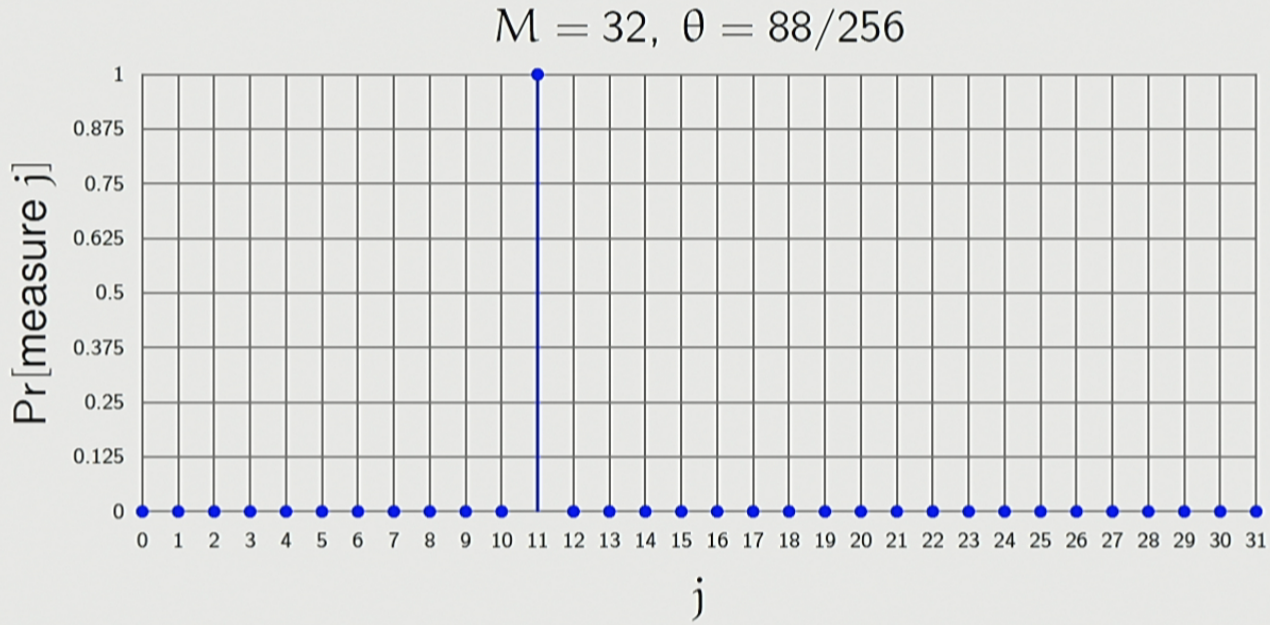
so the probability to measure a given j is

$$\Pr[\text{measure } j] = \frac{1}{M^2} \left| \sum_{k=0}^{M-1} e^{2\pi i k(\theta - j/M)} \right|^2 .$$

This distribution is tightly concentrated around those values of j for which j/M is a close approximation to θ (equating 0 and 1).

Concentration of measurement outcomes

The following plot shows the concentration of measurement outcomes around θM for $M = 32$ and varying values of $\theta \in [10/32, 11/32]$.



In general, after a small number of samples the median j/M value is likely to be within $1/(2M)$ of θ.

Part 3

Shor's algorithm for integer factoring

Shor's algorithm

In 1994, Peter Shor (then at AT&T Labs) discovered a **polynomial-time quantum algorithm** for factoring integers. His algorithm requires

$$O\left((\log N)^3\right)$$

steps, which is asymptotically much better than

$$c^{(\log N)^{1/3}(\log\log N)^{2/3}}$$

steps for the number field sieve.

Shor's algorithm can be understood through the technique of **eigenvalue estimation**, so we've already covered the main quantum part of the algorithm.

Order-finding and factoring

Given an integer $N \geqslant 2$, and another integer a with $\gcd(a, N) = 1$, the **order** of a modulo N is the smallest positive integer r such that

$$a^r \equiv 1 \pmod{N}.$$

The **order-finding problem** is to compute r given N and a as input.

We can hope to factor N by knowing the order of random choices a modulo N. Find nontrivial factors (and then recurse) using this method:

1. Choose $a \in \{2, \ldots, N-1\}$ at random. We're done if $\gcd(a, N) > 1$.
2. Compute the order r of a modulo N.
3. If we are lucky, r is even, so N divides $\left(a^{r/2}+1\right)\left(a^{r/2}-1\right)$. If we are lucky again, $\gcd(a^{r/2}-1, N)$ is a nontrivial factor of N.

Elementary number theory: we will be lucky (both times) in step 3 with probability at least $1/2$ (assuming N is odd and not a prime power).

Order-finding and eigenvalue estimation

For a given choice of N and a with $\gcd(a, N) = 1$, define a unitary transformation

$$M_a |x\rangle = \begin{cases} |ax \pmod N\rangle & \text{if } x \in \{0, \ldots, N-1\} \\ |x\rangle & \text{otherwise.} \end{cases}$$

What are the eigenvalues/eigenvectors of M_a?

Here is one eigenvector with eigenvalue 1:

$$|\psi_0\rangle = |1\rangle + |a\rangle + |a^2\rangle + \cdots + |a^{r-1}\rangle .$$

(Assume everything inside the kets is taken modulo N from now on.)

Here is an eigenvector with eigenvalue $\omega = \exp(2\pi i/r)$:

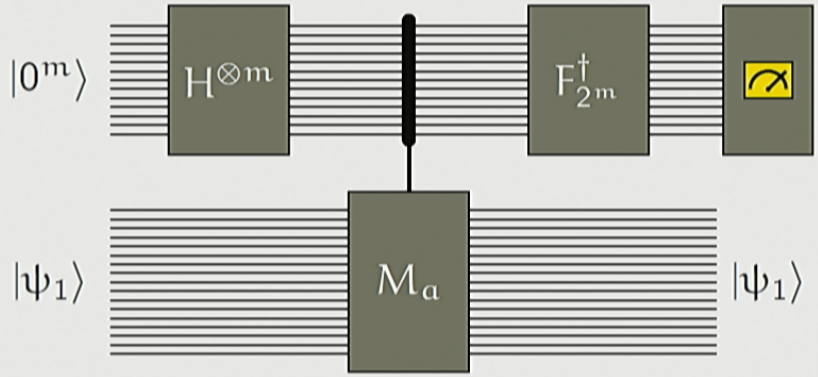
$$|\psi_1\rangle = |1\rangle + \omega^{-1} |a\rangle + \omega^{-2} |a^2\rangle + \cdots + \omega^{-(r-1)} |a^{r-1}\rangle .$$

Order-finding with a nice eigenvector

Suppose that we feed the eigenvector

$$|\psi_1\rangle = \frac{1}{\sqrt{r}}\left(|1\rangle + \omega^{-1}|a\rangle + \omega^{-2}|a^2\rangle + \cdots + \omega^{-(r-1)}|a^{r-1}\rangle\right)$$

into the phase estimation procedure from earlier:



The associated eigenvalue is $\omega = e^{2\pi i\theta}$ for $\theta = 1/r$. After a few repetitions, we obtain $j \in \{0, \ldots, 2^m - 1\}$ so that $j/2^m \approx 1/r$.

Order-finding with a nice eigenvector

We know j so that $j/2^m \approx 1/r$, so rounding $2^m/j$ to the nearest integer will give r if the approximation is close enough. If we take

$$m = 2\lceil \log(N) \rceil + 1, \tag{1}$$

we will have enough precision to recover r.

There are two issues that need to be resolved:

1. We need an efficient implementation of $\Lambda_m(M_a)$ for m as in (1).
2. We do not know how to construct $|\psi_1\rangle$.

The first issue is the easier one: the transformation

$$\Lambda_m(M_a) : |k\rangle\,|x\rangle = |k\rangle\,|a^k x \pmod N\rangle$$

(for $x \in \{0, \ldots, N-1\}$) is **modular exponentiation**, implementable through repeated squaring using $O((\log m)(\log N)^2)$ gates.

Obtaining useful eigenvectors

The second issue, which is that we do not know how to construct $|\psi_1\rangle$, is more serious... we will not try to construct this state, but instead will use an alternate solution.

Let us define

$$|\psi_k\rangle = \frac{1}{\sqrt{r}}\Big(|1\rangle + \omega^{-k}\,|a\rangle + \omega^{-2k}\,|a^2\rangle + \cdots \omega^{-(r-1)k}\,|a^{r-1}\rangle\Big)$$

for each $k = 0, 1, \ldots, r-1$. It holds that

$$M_a\,|\psi_k\rangle = \omega^k\,|\psi_k\rangle\,;$$

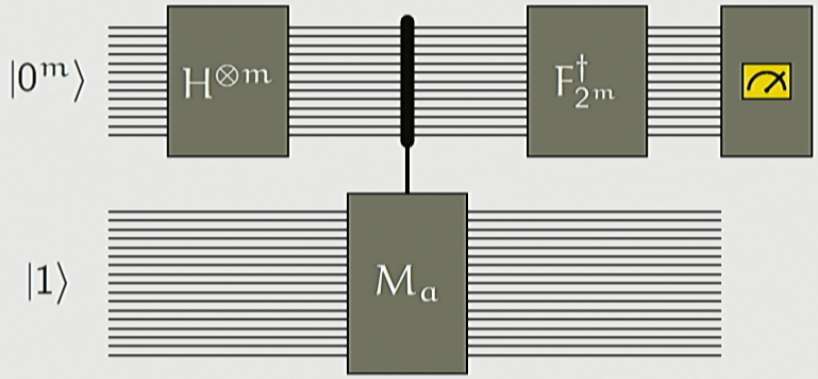
$|\psi_k\rangle$ is an eigenvector of M_a with associated eigenvalue $\omega^k = e^{2\pi i \frac{k}{r}}$.

Let us also observe that

$$|1\rangle = \frac{1}{\sqrt{r}}\big(|\psi_0\rangle + |\psi_1\rangle + \cdots + |\psi_{r-1}\rangle\big).$$

The final eigenvalue estimation procedure

Consider what happens when we substitute $|1\rangle$ for an eigenvector in our eigenvalue estimation procedure:



Owing to the fact that

$$|1\rangle = \frac{1}{\sqrt{r}}\big(|\psi_0\rangle + |\psi_1\rangle + \cdots + |\psi_{r-1}\rangle\big),$$

we will obtain an outcome j with $j/2^m \approx k/r$ for $k \in \{0, 1, \ldots, r-1\}$ chosen **uniformly at random**.

Processing results to obtain the order

Given $j \in \{0, \ldots, 2^m - 1\}$ such that

$$\frac{j}{2^m} \approx \frac{k}{r}$$

for $k \in \{0, 1, \ldots, r-1\}$, we can recover integers s and t so that

$$\frac{s}{t} = \frac{k}{r}$$

using the **continued fraction algorithm**. We then have that t is a divisor of r.

Repeating the entire process a small number of times, we obtain

$$\frac{s_1}{t_1} = \frac{k_1}{r}, \quad \frac{s_2}{t_2} = \frac{k_2}{r}, \quad \frac{s_3}{t_3} = \frac{k_3}{r}, \quad \ldots$$

Summary of Shor's algorithm

To factor an odd, non-prime power integer $N \geqslant 2$, we apply the following algorithm recursively:

1. Choose $a \in \{2, \ldots, N-1\}$ at random. If $d = \gcd(a, N) > 1$, then output d and stop.
2. Compute the order of a modulo N:
   (a) Apply the eigenvalue estimation procedure to the unitary transformation M_a and the state $|1\rangle$, obtaining $j/2^m \approx k/r$ for a random $k \in \{0, \ldots, r-1\}$.
   (b) Use the continued fraction algorithm to compute the fraction s/t with $t < N$ that most closely approximates $j/2^m$.
   (c) Repeat (a) and (b) $O(1)$ times, and let r be the least common multiple of the values of t obtained.
3. If r is even and $d = \gcd(a^{r/2} - 1, N)$ is a proper factor of N, then output d and stop. Otherwise, return to step 1 (or admit failure).

Further reading

Other quantum algorithms, with a variety of speed-ups over classical algorithms are known.

For a comprehensive list, see the **quantum algorithms zoo**:

http://math.nist.gov/quantum/zoo/

If you wish to know more, a good place to start is with these survey papers on quantum algorithms:

- A. Childs and W. van Dam. Quantum algorithms for algebraic problems. *Reviews of Modern Physics* 82: 1–52, 2010. (Available as arXiv:0812.0380.)
- M. Mosca. Quantum Algorithms. In *Encyclopedia of Complexity and Systems Science*, Springer, 2009. (Available as arXiv:0808.0369.)