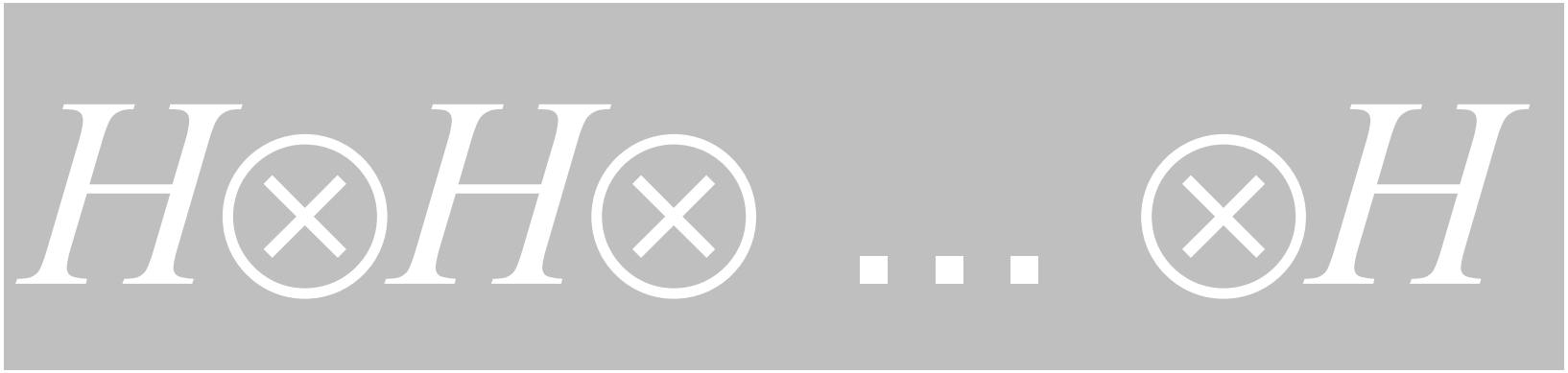
Introduction to Quantum Information Processing

QIC 710 / CS 678 / PH 767 / CO 681 / AM 871

Lectures 6-8 (2013)

Richard Cleve
DC 2117 / QNC 3129
cleve@cs.uwaterloo.ca

Recap from last class …



About $H \otimes H \otimes \ldots \otimes H = H^{\otimes n}$

Theorem: for $x \in \{0,1\}^n$, $H^{\otimes n}|x\rangle = \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$

where $x \cdot y = x_1 y_1 \oplus \ldots \oplus x_n y_n$

Example: $H \otimes H = \frac{1}{2} \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$

Pf: For all $x \in \{0,1\}^n$, $H|x\rangle = |0\rangle + (-1)^x |1\rangle = \Sigma_y (-1)^{xy} |y\rangle$

Thus, $H^{\otimes n}|x_1 \ldots x_n\rangle = \left(\Sigma_{y_1} (-1)^{x_1 y_1} |y_1\rangle\right) \ldots \left(\Sigma_{y_n} (-1)^{x_n y_n} |y_n\rangle\right)$

$= \Sigma_y (-1)^{x_1 y_1 \oplus \ldots \oplus x_n y_n} |y_1 \ldots y_n\rangle$ ∎

Simon’s problem

Quantum vs. classical separations

black-box problem	quantum	classical	
constant vs. balanced	**1** (query)	**2** (queries)	
1-out-of-4 search	**1**	**3**	
constant vs. balanced	**1**	**½ 2^n + 1**	**(only for exact)**
Simon's problem			**(probabilistic)**

Simon's problem

Let $f: \{0,1\}^n \rightarrow \{0,1\}^n$ have the property that there exists an $r \in \{0,1\}^n$ such that $f(x) = f(y)$ iff $x \oplus y = r$ or $x = y$

Example:

x	$f(x)$
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

What is r is this case? _____

Answer: r = 101

A classical algorithm for Simon

Search for a ***collision***, an $x \neq y$ such that $f(x) = f(y)$

1. Choose $x_1, x_2, ..., x_k \in \{0,1\}^n$ randomly (independently)

2. For all $i \neq j$, if $f(x_i) = f(x_j)$ then output $x_i \oplus x_j$ and halt

A hard case is where r is chosen randomly from $\{\mathbf{0},\mathbf{1}\}^n - \{\mathbf{0}^n\}$ and then the "table" for f is filled out randomly subject to the structure implied by r

How big does k have to be for the probability of a collision to be a constant, such as ¾?

Answer: order $2^{n/2}$ (each (x_i, x_j) collides with prob. $O(2^{-n})$)

Classical lower bound

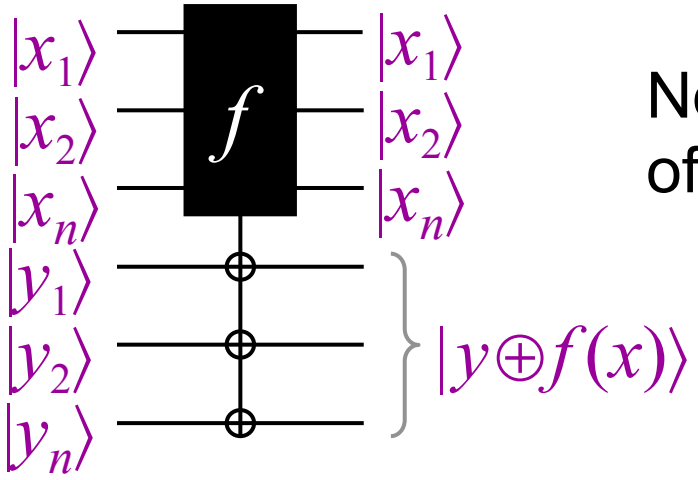
Theorem: ***any*** classical algorithm solving Simon's problem must make $\Omega(2^{n/2})$ queries

Proof is omitted here—note that the performance analysis of the previous algorithm does ***not*** imply the theorem

… how can we know that there isn't a ***different*** algorithm that performs better?

A *quantum* algorithm for Simon I

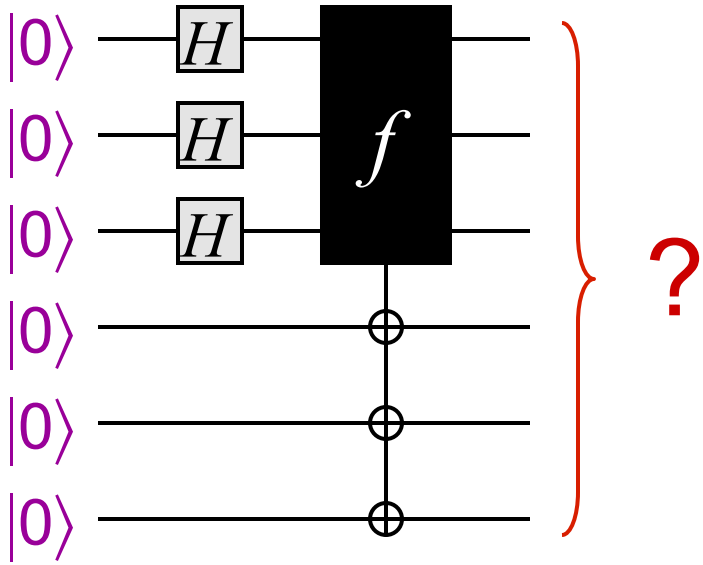
Queries:



Not clear what ***eigenvector*** of target registers is ...

Proposed start of quantum algorithm: query all values of f in superposition

What is the output state of this circuit?



A quantum algorithm for Simon II

Answer: the output state is $\sum_{x\in\{0,1\}^n} |x\rangle|f(x)\rangle$

Let $T \subseteq \{\mathbf{0},\mathbf{1}\}^n$ be such that ***one*** element from each matched pair is in T (assume $r \neq$ **00...0**)

Example: could take $T = \{$**000**, **001**, **011**, **111**$\}$

Then the output state can be written as:

$$\sum_{x\in T} |x\rangle|f(x)\rangle + |x\oplus r\rangle|f(x\oplus r)\rangle$$

$$= \sum_{x\in T} \left(|x\rangle + |x\oplus r\rangle\right)|f(x)\rangle$$

x	$f(x)$
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

A quantum algorithm for Simon III

Measuring the second register yields $|x\rangle + |x \oplus r\rangle$ in the first register, for a random $x \in T$

How can we use this to obtain ***some*** information about r?

Try applying $H^{\otimes n}$ to the state, yielding:

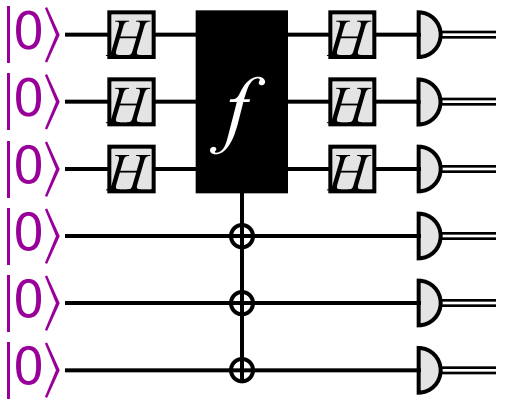
$$\sum_{y \in \{0,1\}^n} (-1)^{x \bullet y} |y\rangle + \sum_{y \in \{0,1\}^n} (-1)^{(x \oplus r) \bullet y} |y\rangle$$

$$= \sum_{y \in \{0,1\}^n} (-1)^{x \bullet y} \left(1 + (-1)^{r \bullet y}\right) |y\rangle$$

Measuring this state yields y with prob. $\begin{cases} (1/2)^{n-1} & \text{if } r \cdot y = 0 \\ 0 & \text{if } r \cdot y \neq 0 \end{cases}$

A quantum algorithm for Simon IV

Executing this algorithm $k = O(n)$ times yields random $y_1, y_2, ..., y_k \in \{0,1\}^n$ such that $r \cdot y_1 = r \cdot y_2 = ... = r \cdot y_n = 0$



How does this help?

This is a system of k linear equations:

$$\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{k1} & y_{k2} & \cdots & y_{kn} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

With high probability, there is a unique non-zero solution that is r (which can be efficiently found by linear algebra)

Conclusion of Simon's algorithm

- Any classical algorithm has to query the black box $\Omega(2^{n/2})$ times, even to succeed with probability ¾
- There is a quantum algorithm that queries the black box only $O(n)$ times, performs only $O(n^3)$ auxiliary operations (for the Hadamards, measurements, and linear algebra), and succeeds with probability ¾

Discrete log problem

Discrete logarithm problem (DLP)

Input: p (prime), g (generator of $\mathbf{Z}^*_p$), $a \in \mathbf{Z}^*_p$

Output: $r \in \mathbf{Z}_{p-1}$ such that $g^r \bmod p = a$

Example: $p = 7$, $\mathbf{Z}^*_7 = \{1, 2, 3, 4, 5, 6\} = \{3^0, 3^2, 3^1, 3^4, 3^5, 3^3\}$
(hence 3 is a generator of $\mathbf{Z}^*_7$)

For $a = 6$, since $3^3 = 6$, the output should be $r = 3$

Note: No efficient classical algorithm for ***DLP*** is known (and cryptosystems exist whose security is predicated on the computational difficulty of DLP)

Efficient quantum algorithm for DLP?
(**Hint:** it can be made to look like Simon's problem!)

DLP similar to Simon's problem

Clever idea (of Shor): define $f : \mathbf{Z}_{p-1} \times \mathbf{Z}_{p-1} \rightarrow \mathbf{Z}^*_p$ as $f(x_1, x_2) = g^{x_1} a^{-x_2} \bmod p$ (can be efficiently computed)

When is $f(x_1, x_2) = f(y_1, y_2)$?

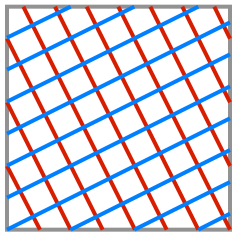
We know $a = g^r$ for ***some*** r, so $f(x_1, x_2) = g^{x_1 - r x_2} \bmod p$

Thus, $f(x_1, x_2) = f(y_1, y_2)$ iff $x_1 - r x_2 \equiv y_1 - r y_2 \pmod{p-1}$

iff $(x_1, x_2) \cdot (1, -r) \equiv (y_1, y_2) \cdot (1, -r) \pmod{p-1}$

iff $((x_1, x_2) - (y_1, y_2)) \cdot (1, -r) \equiv 0 \pmod{p-1}$

iff $(x_1, x_2) - (y_1, y_2) \equiv k(r, 1) \pmod{p-1}$



$\mathbf{Z}_{p-1} \times \mathbf{Z}_{p-1}$

Recall Simon's property: $f(x) = f(y)$ iff $x - y = kr \pmod 2$

Introduction to Quantum Information Processing

QIC 710 / CS 678 / PH 767 / CO 681 / AM 871

Lecture 7 (2013)

Richard Cleve
DC 2117 / QNC 3129
cleve@cs.uwaterloo.ca

Recap from last class …

Discrete logarithm problem (DLP)

Input: p (prime), g (generator of $\mathbf{Z}^*_p$), $a \in \mathbf{Z}^*_p$

Output: $r \in \mathbf{Z}_{p-1}$ such that $g^r \bmod p = a$

Example: $p = 7$, $\mathbf{Z}^*_7 = \{1, 2, 3, 4, 5, 6\} = \{3^0, 3^2, 3^1, 3^4, 3^5, 3^3\}$
(hence 3 is a generator of $\mathbf{Z}^*_7$)

For $a = 6$, since $3^3 = 6$, the output should be $r = 3$

Note: No efficient classical algorithm for ***DLP*** is known (and cryptosystems exist whose security is predicated on the computational difficulty of DLP)

Efficient quantum algorithm for DLP?
(**Hint:** it can be made to look like Simon's problem!)

DLP similar to Simon's problem

Clever idea (of Shor): define $f : \mathbf{Z}_{p-1} \times \mathbf{Z}_{p-1} \rightarrow \mathbf{Z}^*_p$ as $f(x_1, x_2) = g^{x_1} a^{-x_2} \bmod p$ (can be efficiently computed)

When is $f(x_1, x_2) = f(y_1, y_2)$?

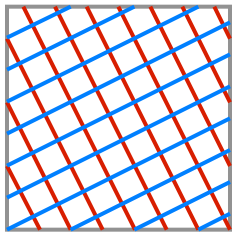
We know $a = g^r$ for ***some*** r, so $f(x_1, x_2) = g^{x_1 - r x_2} \bmod p$

Thus, $f(x_1, x_2) = f(y_1, y_2)$ iff $x_1 - r x_2 \equiv y_1 - r y_2 \pmod{p-1}$

iff $(x_1, x_2) \cdot (1, -r) \equiv (y_1, y_2) \cdot (1, -r) \pmod{p-1}$

iff $((x_1, x_2) - (y_1, y_2)) \cdot (1, -r) \equiv 0 \pmod{p-1}$

iff $(x_1, x_2) - (y_1, y_2) \equiv k(r, 1) \pmod{p-1}$



Recall Simon's property: $f(x) = f(y)$ iff $x - y = kr \pmod 2$

Simon's problem modulo m

The function arising in DLP can be abstracted to the following

Given: $f : \mathbf{Z}_m \times \mathbf{Z}_m \rightarrow \mathbf{T}$ with the property that:

$f(x_1, x_2) = f(y_1, y_2)$ iff $(x_1, x_2) - (y_1, y_2) \equiv k(r_1, r_2) \pmod m$

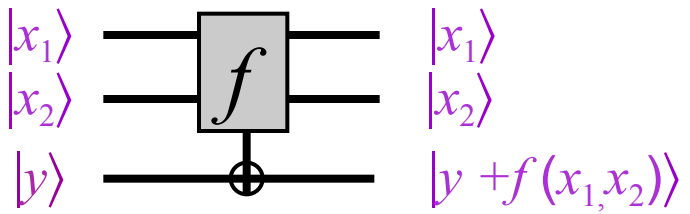
where (r_1, r_2) is the hidden data

Goal: determine (r_1, r_2)

Note: in DLP case, $(r_1, r_2) = (r, 1)$

The reversible query box for f is:

(Each "wire" denotes several qubit wires, enough to represent elements of $\mathbf{Z}_m$)



It's not a "black" box, because we can simulate it by 1- and 2-qubit gates (and this can be done efficiently) …

Digression:
on simulating black boxes

How *not* to simulate a black box

Given an explicit function, such as $f(x) = g^{x_1} a^{-x_2} \bmod p$, over some finite domain, simulate f-queries over that domain

Easy to compute mapping $|x\rangle|y\rangle|00...0\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle|g(x)\rangle$, where the third register is "work space" with accumulated "garbage" (e.g., two such bits arise when a Toffoli gate is used to simulate an AND gate)

This works fine – ***as long as f is not queried in superposition***

If f is queried in superposition then the resulting state can be $\sum_x \alpha_x |x\rangle|y \oplus f(x)\rangle|g(x)\rangle$ can we just discard the third register?

No ... there could be entanglement ...

How *to* simulate a black box

Simulate the mapping $|x\rangle|y\rangle|00...0\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle|00...0\rangle$,
(i.e., clean up the "garbage")

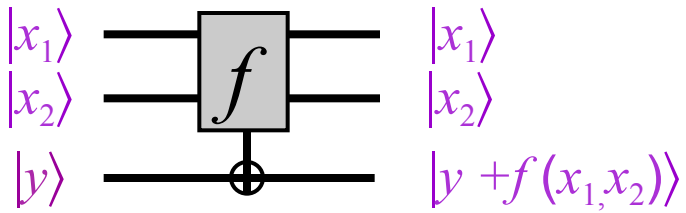
To do this, use an additional register, and:

1. compute $|x\rangle|y\rangle|00...0\rangle|00...0\rangle \rightarrow |x\rangle|y\rangle|f(x)\rangle|g(x)\rangle$
   (ignoring the 2nd register in this step)
2. compute $|x\rangle|y\rangle|f(x)\rangle|g(x)\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle|f(x)\rangle|g(x)\rangle$
   (using CNOT gates between the 2nd and 3rd registers)
3. compute $|x\rangle|y \oplus f(x)\rangle|f(x)\rangle|g(x)\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle|00...0\rangle|00...0\rangle$
   (by reversing the procedure in step 1)

Total cost: around twice the classical cost of computing f, plus n auxiliary gates

Simon's problem modulo m

So now we have an efficient way of implementing the reversible black box for f



Each "wire" denotes several qubits, to represent an element of $\mathbf{Z}_m$

What is a quantum algorithm for this problem? To get one, must go beyond the Hadamard transform, which has been our main tool so far, to …

Quantum Fourier transform (QFT)

Quantum Fourier transform

$$F_m = \frac{1}{\sqrt{m}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{m-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(m-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(m-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{m-1} & \omega^{2(m-1)} & \omega^{3(m-1)} & \cdots & \omega^{(m-1)^2} \end{bmatrix}$$

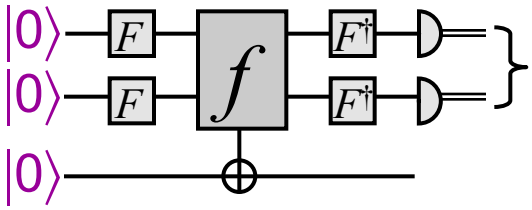
where $\omega = e^{2\pi i/m}$ (for n qubits, $m = 2^n$)

This is unitary and $F_2 = H$, the Hadamard transform

This generalization of H is an important component of several interesting quantum algorithms …

Quantum algorithm for Simon mod m

$f(x_1, x_2) = f(y_1, y_2)$ iff $(x_1, x_2) - (y_1, y_2) \equiv k(r, 1) \pmod{m}$



the result is a random (s_1, s_2) such that

$(s_1, s_2) \cdot (r, 1) \equiv 0 \pmod{m}$

Assignment #3

if $\gcd(s_1, m) = 1$ then s_1 has an inverse mod m, and r can be computed as $r = -s_2 / s_1 \bmod m$
(The details follow from the extended Euclidean algorithm)

Moreover, the probability that $\gcd(s_1, m) = 1$ occurs is not too small (if it fails the algorithm can be run again)

Quantum algorithm for Simon mod m

Steps that have been shown to be efficiently implementable (i.e., in terms of a number of 1- and 2-qubit/bit gates that scales polynomially with respect to the number of bits of m):

- Implementation of reversible gate for f
- The classical post-processing at the end

What's missing? The implementation of the QFT f modulo m (for DLP, we would need to do this for $m = p - 1$)

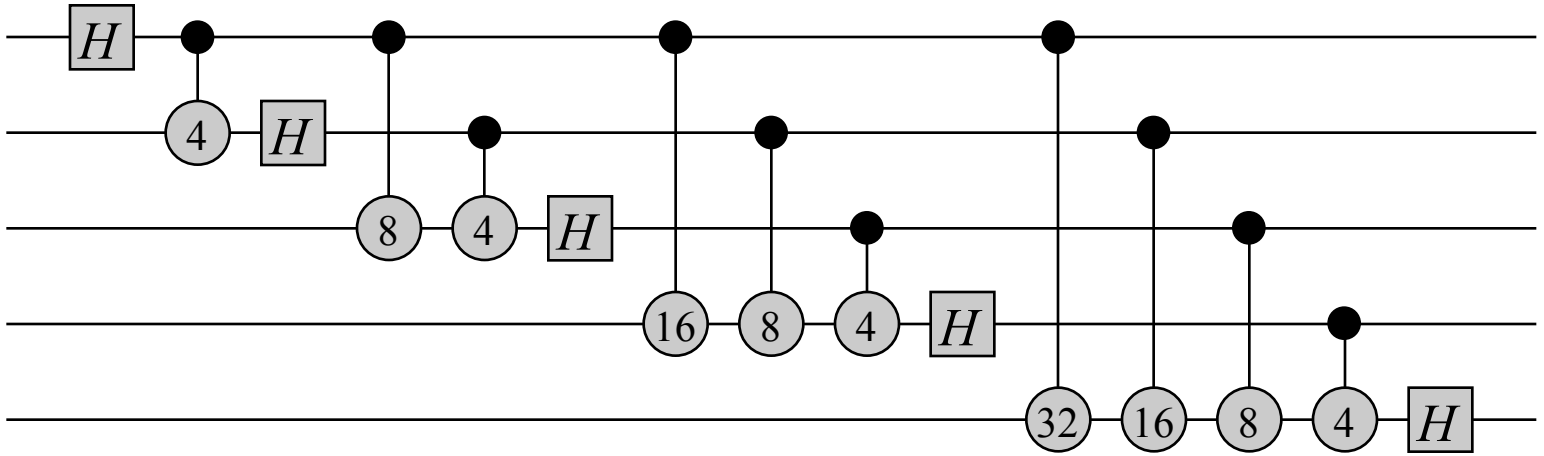
We'll just show how to implement the QFT for $m = 2^n$

Shor did thus too, and showed that if the modulus is within a factor of 2 from $p - 1$, by using careful error-analysis, this was good enough, though the calculations and analysis become more complicated (we omit the details of this)

Computing the QFT for $m = 2^n$

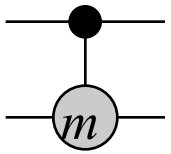
Computing the QFT for $m = 2^n$ (1)

Quantum circuit for F_{32}:



reverse order

Gates: $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$



$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/m} \end{bmatrix}$$

For F_{2^n} costs $O(n^2)$ gates

Computing the QFT for $m = 2^n$ (2)

One way on seeing why this circuit works is to first note that

$$F_{2^n}|a_1 a_2 \ldots a_n\rangle$$

$$= (|0\rangle + e^{2\pi i(0.a_n)}|1\rangle)\ldots(|0\rangle + e^{2\pi i(0.a_2 \ldots a_n)}|1\rangle)(|0\rangle + e^{2\pi i(0.a_1 a_2 \ldots a_n)}|1\rangle)$$

It can then be checked that the circuit produces these states (with qubits in reverse order) for all computational basis states $|a_1 a_2 \ldots a_n\rangle$

Exercise: (a) prove the above equation from the definition of the QFT; (b) confirm that the circuit produces these states

Introduction to Quantum Information Processing

QIC 710 / CS 678 / PH 767 / CO 681 / AM 871

Lecture 8 (2013)

Richard Cleve
DC 2117 / QNC 3129
cleve@cs.uwaterloo.ca

Continuing with the QFT for $m = 2^n$

Quantum Fourier transform

$$F_m = \frac{1}{\sqrt{m}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{m-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(m-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(m-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{m-1} & \omega^{2(m-1)} & \omega^{3(m-1)} & \cdots & \omega^{(m-1)^2} \end{bmatrix}$$

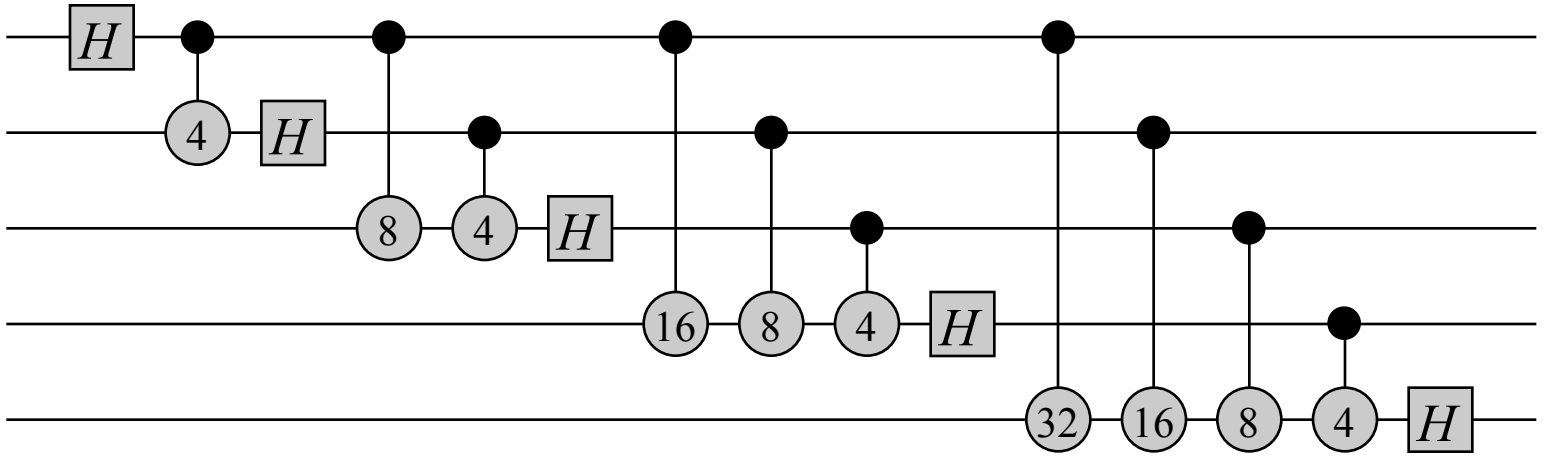
where $\omega = e^{2\pi i/m}$ (for n qubits, $m = 2^n$)

This is unitary and $F_2 = H$, the Hadamard transform

This generalization of H is an important component of several interesting quantum algorithms …

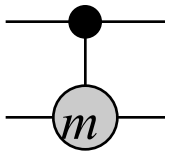
Computing the QFT for $m = 2^n$ (1)

Quantum circuit for F_{32}:



reverse order

Gates: $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$



$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/m} \end{bmatrix}$$

For F_{2^n} costs $O(n^2)$ gates

Computing the QFT for $m = 2^n$ (2)

One way on seeing why this circuit works is to first note that

$$F_{2^n}|a_1 a_2 \ldots a_n\rangle$$

$$= (|0\rangle + e^{2\pi i(0.a_n)}|1\rangle)\ldots(|0\rangle + e^{2\pi i(0.a_2 \ldots a_n)}|1\rangle)(|0\rangle + e^{2\pi i(0.a_1 a_2 \ldots a_n)}|1\rangle)$$

It can then be checked that the circuit produces these states (with qubits in reverse order) for all computational basis states $|a_1 a_2 \ldots a_n\rangle$

Exercise: (a) prove the above equation from the definition of the QFT; (b) confirm that the circuit produces these states

Hidden Subgroup Problem framework

Hidden subgroup problem (1)

Let G be a known group and H be an unknown subgroup of G

Let $f: G \rightarrow T$ have the property $f(x) = f(y)$ iff $x - y \in H$
(i.e., x and y are in the same ***right coset*** of H)

Problem: given a black-box for computing f, determine H

Example 1: $G = (\mathbf{Z}_2)^n$ (the additive group) and $H = \{0, r\}$

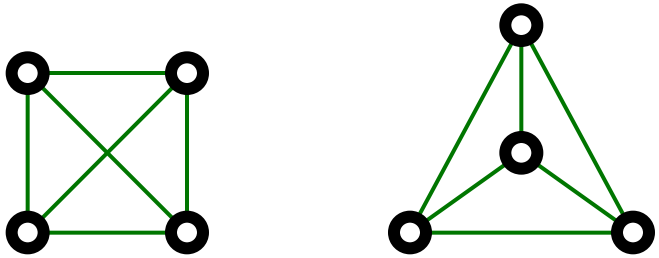
Example 2: $G = (\mathbf{Z}_{p-1})^2$ and
$H = \{(0,0), (r,1), (2r,2), \ldots, ((p-2)r, p-2)\}$

Example 3: $G = \mathbf{Z}$ and $H = r\mathbf{Z}$ (Shor's factoring algorithm was originally approached this way. A complication that arises is that **Z** is infinite. We'll use a different approach)

Hidden subgroup problem (2)

Example 4: $G = S_n$ (the symmetric group, consisting of all permutations on n objects—which is not abelian) and H is any subgroup of G

A quantum algorithm for this instance of HSP ***would*** lead to an efficient quantum algorithm for the graph isomorphism problem …



… ***alas*** no efficient quantum has been found for this instance of HSP, despite significant effort by many people

Eigenvalue estimation problem (a.k.a. phase estimation)

Note: this will lead to a factoring algorithm similar to Shor's

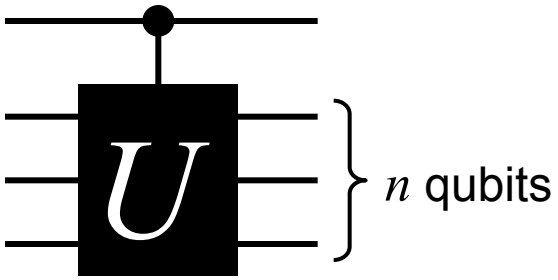
A simplified example

U is an unknown unitary operation on n qubits

$|\psi\rangle$ is an eigenvector of U, with eigenvalue $\lambda = +1 \text{ or } -1$

Input: a black-box for a controlled-U

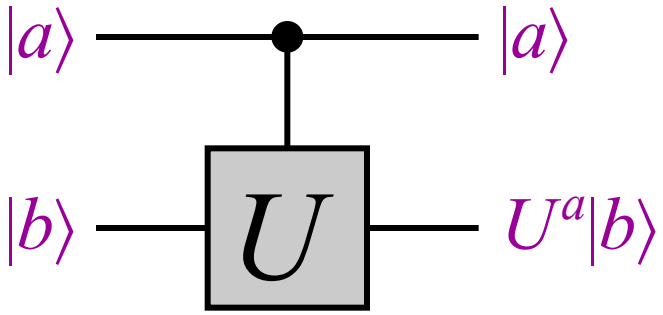
and a copy of the state $|\psi\rangle$



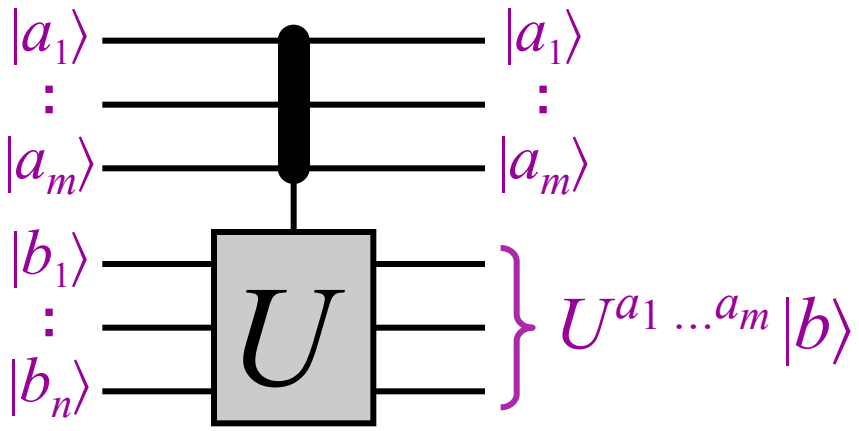
Output: the eigenvalue λ

Exercise: solve this making a single query to the controlled-U

Generalized controlled-U gates



$$\begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix}$$



$$\begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ 0 & U & 0 & \cdots & 0 \\ 0 & 0 & U^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & U^{2^m-1} \end{bmatrix}$$

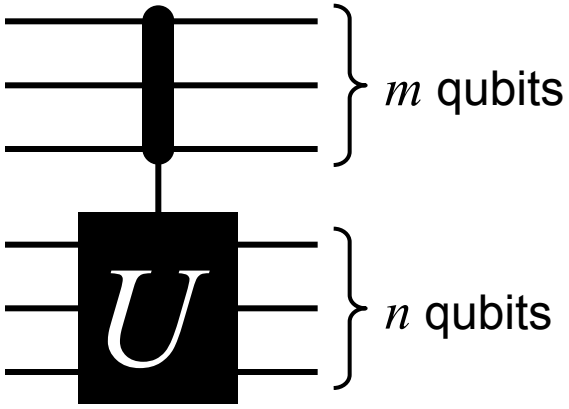
Example: $|1101\rangle|0101\rangle \rightarrow |1101\rangle U^{1101}|0101\rangle$

Eigenvalue estimation problem

U is a unitary operation on n qubits

$|\psi\rangle$ is an eigenvector of U, with eigenvalue $e^{2\pi i\phi}$
$(0 \leq \phi < 1)$

Input: black-box for

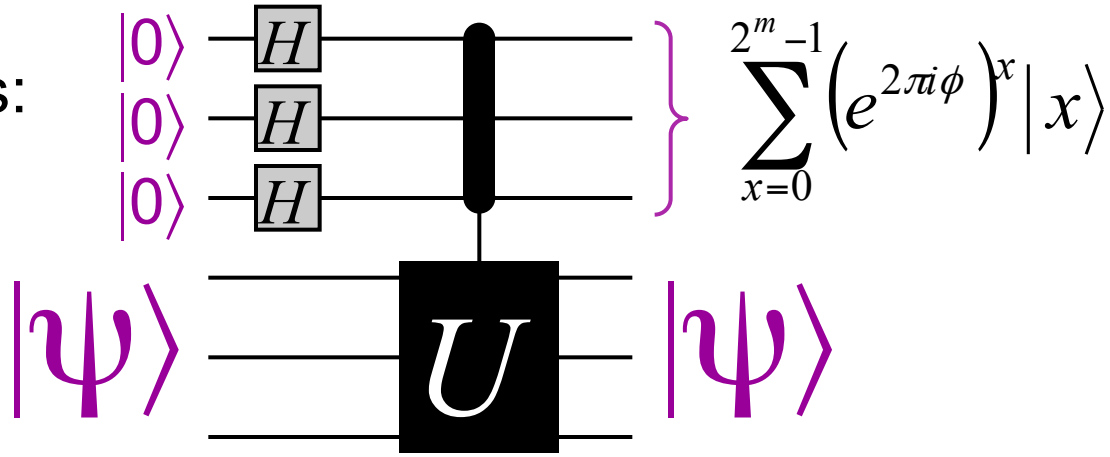


and a copy of $|\psi\rangle$

Output: ϕ (m-bit approximation)

Algorithm for eigenvalue estimation (1)

Starts off as:



$|00\ldots0\rangle|\psi\rangle$

$|a\rangle|b\rangle \rightarrow |a\rangle U^a|b\rangle$

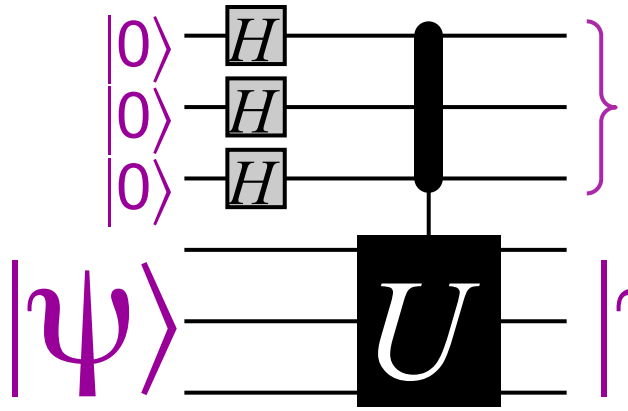
$\rightarrow (|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\ldots(|0\rangle + |1\rangle)|\psi\rangle$

$= (|000\rangle + |001\rangle + |010\rangle + |011\rangle + \ldots + |111\rangle)|\psi\rangle$

$= (|0\rangle + |1\rangle + |2\rangle + |3\rangle + \ldots + |2^m - 1\rangle)|\psi\rangle$

$\rightarrow (|0\rangle + e^{2\pi i\phi}|1\rangle + (e^{2\pi i\phi})^2|2\rangle + (e^{2\pi i\phi})^3|3\rangle + \ldots + (e^{2\pi i\phi})^{2^m-1}|2^m-1\rangle)|\psi\rangle$

Algorithm for eigenvalue estimation (2)



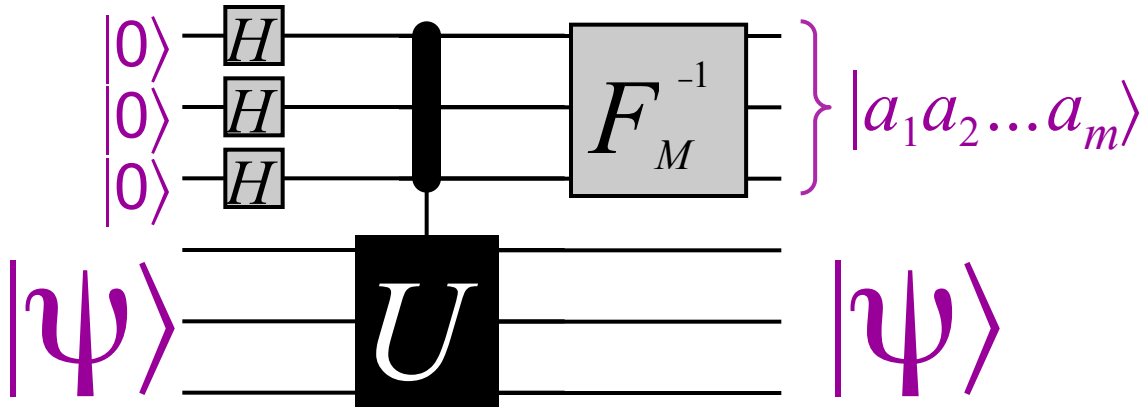
$$\sum_{x=0}^{2^m-1} \left(e^{2\pi i\phi}\right)^x |x\rangle$$

Recall that $F_M |a_1 a_2 \ldots a_m\rangle = \sum_{x=0}^{2^m-1} \left(e^{2\pi i(0.a_1 a_2 \ldots a_m)}\right)^x |x\rangle$

$$F_M^{-1} = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \cdots & \omega^{-(M-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} & \cdots & \omega^{-2(M-1)} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} & \cdots & \omega^{-3(M-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(M-1)} & \omega^{-2(M-1)} & \omega^{-3(M-1)} & \cdots & \omega^{-(M-1)^2} \end{bmatrix}$$

Therefore, when $\phi = 0.a_1 a_2 \ldots a_m$ applying the ***inverse*** of F_M yields ϕ (digits)

Algorithm for eigenvalue estimation (3)



If $\phi = 0.a_1 a_2 \ldots a_m$ then the above procedure yields $|a_1 a_2 \ldots a_m\rangle$ (from which ϕ can be deduced exactly)

But what ϕ if is not of this nice form?

Example: $\phi = ⅓ = 0.0101010101010101\ldots$

Algorithm for eigenvalue estimation (4)

What if ϕ is not of the nice form $\phi = 0.a_1 a_2 \ldots a_m$?

Example: $\phi = \frac{1}{3} = 0.\underline{01010101}01010101\ldots$

Let's calculate what the previously-described procedure does:

Let $a/2^m = 0.a_1 a_2 \ldots a_m$ be an m-bit approximation of ϕ, in the sense that $\phi = a/2^m + \delta$, where $|\delta| \leq 1/2^{m+1}$

$$\left(F_M\right)^{-1} \sum_{x=0}^{2^m-1} \left(e^{2\pi i \phi}\right)^x \left|x\right\rangle = \frac{1}{2^m} \sum_{y=0}^{2^m-1} \sum_{x=0}^{2^m-1} e^{-2\pi i x y / 2^m} e^{2\pi i \phi x} \left|y\right\rangle$$

$$= \frac{1}{2^m} \sum_{y=0}^{2^m-1} \sum_{x=0}^{2^m-1} e^{-2\pi i x y / 2^m} e^{2\pi i \left(\frac{a}{2^m} + \delta\right) x} \left|y\right\rangle$$

$$= \frac{1}{2^m} \sum_{y=0}^{2^m-1} \sum_{x=0}^{2^m-1} e^{2\pi i (a-y) x / 2^m} e^{2\pi i \delta x} \left|y\right\rangle$$

What is the amplitude of $|a_1 a_2 \ldots a_m\rangle$?

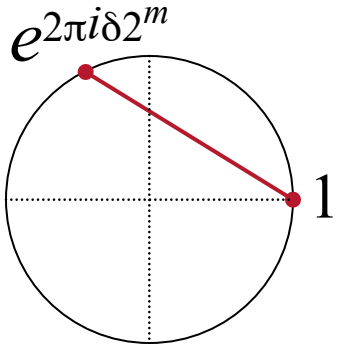
Algorithm for eigenvalue estimation (5)

State is: $\frac{1}{2^m}\sum_{y=0}^{2^m-1}\sum_{x=0}^{2^m-1} e^{2\pi i(a-y)x/2^m} e^{2\pi i\delta x}|y\rangle$

geometric series!

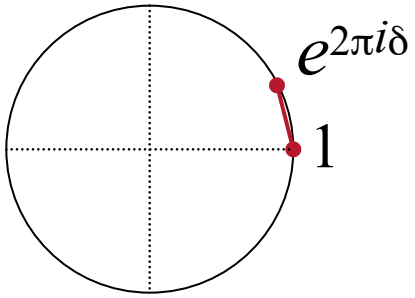
The amplitude of $|y\rangle$, for $y = a$ is $\frac{1}{2^m}\sum_{x=0}^{2^m-1} e^{2\pi i\delta x} = \frac{1}{2^m}\frac{1-\left(e^{2\pi i\delta}\right)^{2^m}}{1-e^{2\pi i\delta}}$

Numerator:



lower bounded by
$2\pi\delta 2^m(2/\pi) > 4\delta 2^m$

Denominator:



upper bounded by $2\pi\delta$

Therefore, the absolute value of the amplitude of $|y\rangle$ is at least the quotient of $(1/2^m)$(numerator/denominator), which is $2/\pi$

Algorithm for eigenvalue estimation (6)

Therefore, the probability of measuring an m-bit approximation of ϕ is always at least $4/\pi^2 \approx 0.4$

For example, when $\phi = \frac{1}{3} = 0.\underline{0101}0101010101\ldots$, the outcome probabilities look roughly like this: