

Introduction to Quantum Information Processing

QIC 710 / CS 768 / PH 767 / CO 681 / AM 871

Lectures 6–8 (2019)

Richard Cleve

QNC 3129

cleve@uwaterloo.ca

Discrete log problem

Discrete logarithm problem (DLP)

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

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

Example: $p = 7$, $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\} = \{3^0, 3^2, 3^1, 3^4, 3^5, 3^3\}$
(hence 3 is a generator of \mathbb{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 $f: \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \rightarrow \mathbb{Z}_p^*$ as
 $f(x_1, x_2) = g^{x_1} a^{-x_2} \pmod p$ (can be efficiently computed)

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

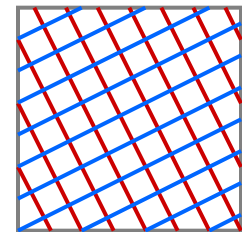
We know $a = g^r$ for **some** r , so $f(x_1, x_2) = g^{x_1 - rx_2} \pmod p$

Thus, $f(x_1, x_2) = f(y_1, y_2)$ iff $x_1 - rx_2 \equiv y_1 - ry_2 \pmod{p-1}$

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



$(1, -r)$

$(r, 1)$

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

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

Simon's problem modulo m

The function arising in DLP can be abstracted to the following

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

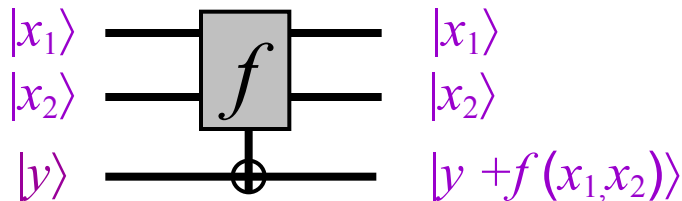
$$f(x_1, x_2) = f(y_1, y_2) \text{ iff } (x_1, x_2) - (y_1, y_2) \equiv k(r_1, r_2) \pmod{m}$$

where (r_1, r_2) is the hidden data

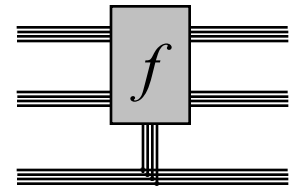
Goal: determine (r_1, r_2)

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

The reversible query box for f is:



where each “wire” denotes many qubit wires, to represent elements of \mathbb{Z}_m like:



Not a “black” box, because we can simulate it by 1-qubit and 2-qubit gates (and this can be done efficiently) ...

Digression: on simulating black boxes

How *not* to simulate a black box

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

Easy to compute mapping $|x\rangle|y\rangle|00\dots 0\rangle \mapsto |x\rangle|y \oplus f(x)\rangle|g(x)\rangle$, 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 $|x\rangle|y\rangle|00\dots 0\rangle \mapsto |x\rangle|y\oplus f(x)\rangle|00\dots 0\rangle$,
(i.e., clean up the “garbage”)

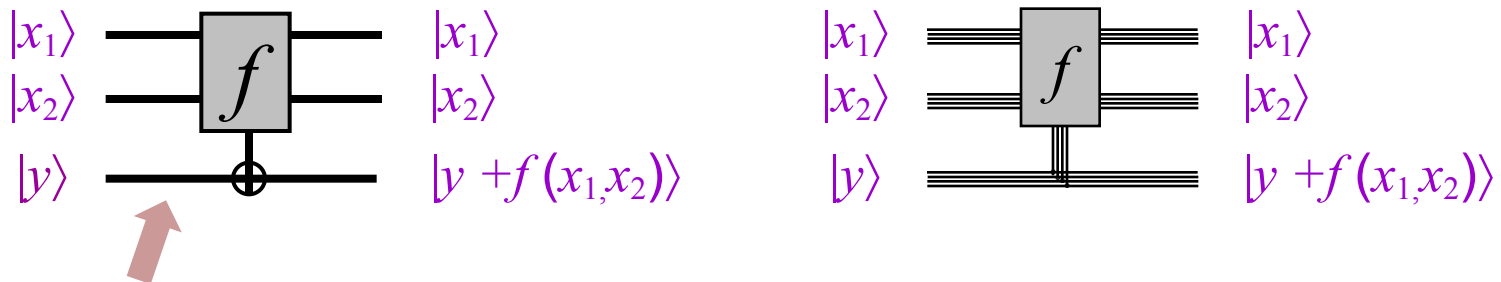
To do this, use an additional register, and:

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

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

Simon's problem modulo m

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



Reminder: each “thick wire” denotes several qubits, to represent an element of \mathbb{Z}_m (eg, $\{0, 1, 2, 3, 4, 5, 6\} = \{000, 001, 010, 011, 100, 101, 110\}$)

OK, so what about a quantum algorithm for this problem?

To get one, we go beyond the Hadamard transform, which has been our main tool so far, to ...

Quantum Fourier transform (QFT)

Quantum Fourier transform

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

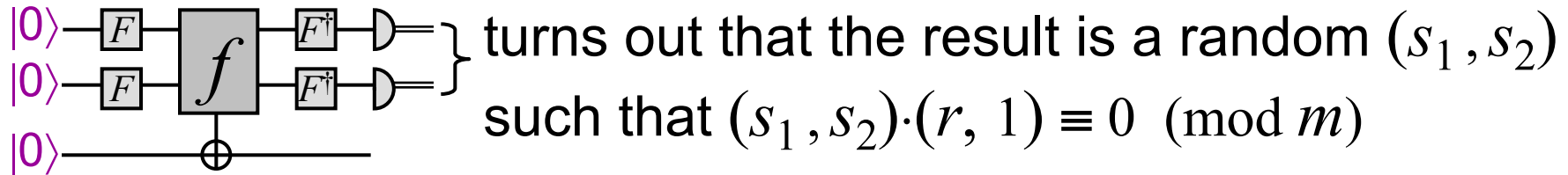
where $\omega = e^{2\pi i/m}$ (for n qubits, $m = 2^n$)

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 (1)

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



The state right after the query is $\frac{1}{m} \sum_{x_1 \in \mathbb{Z}_m} \sum_{x_2 \in \mathbb{Z}_m} |x_1\rangle |x_2\rangle |f(x_1, x_2)\rangle$

Now, if the third register is measured in the computational basis then it collapses to some value, and state of the first two registers is a superposition of all (x_1, x_2) that f maps to that value, which is a state of the form

$$\begin{aligned} & \frac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} |x_1 + kr_1\rangle |x_2 + kr_2\rangle \\ &= \frac{1}{\sqrt{m}} \left(|(x_1, x_2)\rangle + |(x_1, x_2) + (r_1, r_2)\rangle + \cdots + |(x_1, x_2) + (m-1)(r_1, r_2)\rangle \right) \end{aligned}$$

Quantum algorithm for Simon mod m (2)

Here is the state again:

$$\begin{aligned} & \frac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} |x_1 + kr_1\rangle |x_2 + kr_2\rangle \\ &= \frac{1}{\sqrt{m}} \left(|(x_1, x_2)\rangle + |(x_1, x_2) + (r_1, r_2)\rangle + \cdots + |(x_1, x_2) + (m-1)(r_1, r_2)\rangle \right) \end{aligned}$$

The next step is to apply the two inverse Fourier transforms mod m , yielding

$$\begin{aligned} \left(F_m^\dagger \otimes F_m^\dagger \right) \frac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} |x_1 + kr_1\rangle |x_2 + kr_2\rangle &= \frac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} F_m^\dagger |x_1 + kr_1\rangle F_m^\dagger |x_2 + kr_2\rangle \\ &= \frac{1}{m^{3/2}} \sum_{k \in \mathbb{Z}_m} \sum_{s_1 \in \mathbb{Z}_m} \omega^{-s_1(x_1 + kr_1)} |s_1\rangle \sum_{s_2 \in \mathbb{Z}_m} \omega^{-s_2(x_2 + kr_2)} |s_2\rangle \\ &= \frac{1}{\sqrt{m}} \sum_{s_1} \sum_{s_2} \left(\frac{1}{m} \sum_{k \in \mathbb{Z}_m} \omega^{-(s_1, s_2) \cdot ((x_1, x_2) + k(r_1, r_2))} \right) |s_1, s_2\rangle \\ &= \frac{1}{\sqrt{m}} \sum_{s_1, s_2} \omega^{-(s_1, s_2) \cdot (x_1, x_2)} \left(\frac{1}{m} \sum_{k \in \mathbb{Z}_m} \omega^{-(s_1, s_2) \cdot (r_1, r_2)k} \right) |s_1, s_2\rangle \end{aligned}$$

Quantum algorithm for Simon mod m (3)

Note that
$$\frac{1}{m} \sum_{k \in \mathbb{Z}_m} \omega^{-(s_1, s_2) \cdot (r_1, r_2)k} = \begin{cases} 1 & \text{if } (s_1, s_2) \cdot (r_1, r_2) = 0 \\ 0 & \text{otherwise} \end{cases}$$

So the amplitudes of all basis states $|s_1, s_2\rangle$ where $(s_1, s_2) \cdot (r_1, r_2) \neq 0$ are zero

Therefore, if the first two registers are measured, the result is a **random** (s_1, s_2) subject to the condition that it has dot product 0 with (r_1, r_2)

The dot product condition implies that (r_1, r_2) satisfies the linear relationship $s_1 r_1 + s_2 r_2 \equiv 0 \pmod{m}$

As with Simon's problem, we can repeat this process until we have enough linear relationships to deduce (r_1, r_2)

A complication is that, if the modulus m is not prime then we are not working over a *field*, so we are outside the framework of *linear algebra*

For the Discrete Log Problem, $m = p - 1$ (which is not prime) and $(r_1, r_2) = (1, r)$

Quantum algorithm for Simon mod m (4)

In the context of DLP, we have $(s_1, s_2) \cdot (r, 1) \equiv s_1 r + s_2 \equiv 0 \pmod{p-1}$

If s_1 has an inverse then we can solve for r as $r = -s_2/s_1$

In our mod $p-1$ arithmetic, if s_1 and $p-1$ are **coprime** (see below) then s_1 has an inverse mod $p-1$

Moreover, the probability that s_1 and $p-1$ are coprime occurs is not too small (and if it fails on one run then the algorithm can be run again)

Definition: a_1 and a_2 are **coprime** if their largest common divisor is 1 (for example, 12 and 21 are **not** coprime, since 3 is a common divisor, but 10 and 21 are coprime)

Lemma: if a_1 and a_2 are coprime then a_1 has an inverse modulo a_2

Proof idea: the Extended Euclidean Algorithm implies that if a_1 and a_2 are coprime then there exist integers b_1 and b_2 such that $b_1 a_1 + b_2 a_2 = 1$ (e.g., for 10 and 21, we have $(-2)10 + (1)21 = 1$)

This implies that $b_1 a_1 = 1 - b_2 a_2$ so $b_1 a_1 \equiv 1 \pmod{a_2}$

Therefore $b_1 = a_1^{-1} \pmod{a_2}$

Quantum algorithm for Simon mod m (5)

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?

Implementation of the QFT f modulo m ($= p - 1$ for DLP)

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

Shor did this 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)

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

Quantum Fourier transform

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

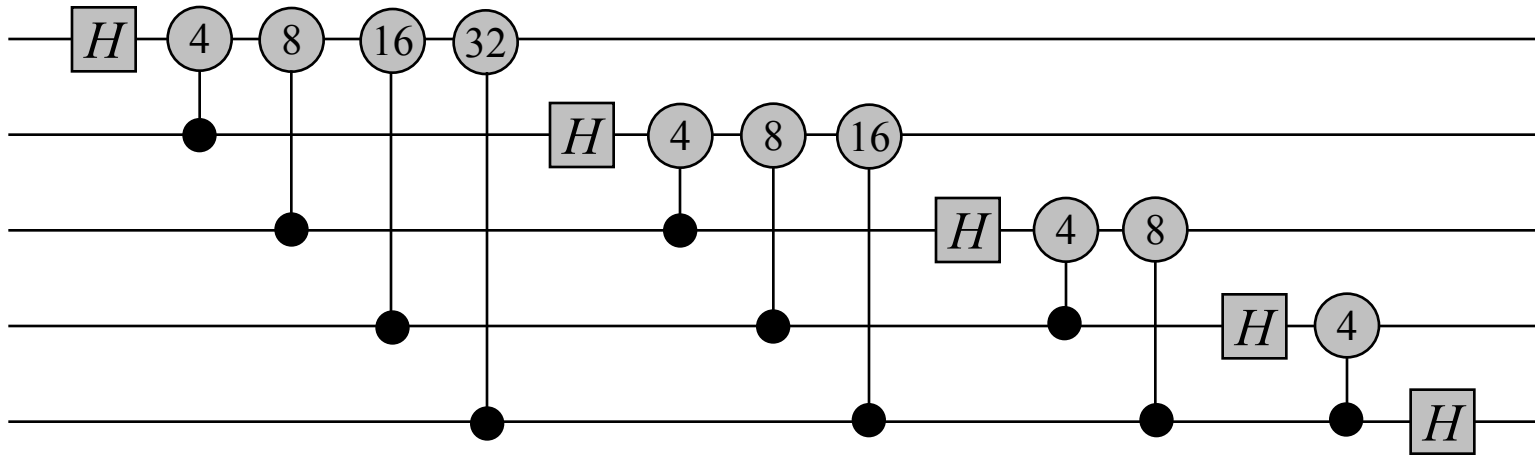
where $\omega = e^{2\pi i/m}$ (for n qubits, $m = 2^n$)

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} :



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

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

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

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

Binary numbers (base-two representation of integers)

We identify $\{000, 001, 010, 011, 100, 101, 110, 111\} = \{0, 1, 2, 3, 4, 5, 6, 7\}$

Formally, for $a = a_1a_2\dots a_n$, define $(a_1a_2\dots a_n)$ to be the corresponding integer

Binary fractions (base-two representation of rational numbers)

What are (0.1) ?, (0.01) , (0.11) ?

As in the base-ten case, shifting the radix point left by is equivalent to dividing by the base number

Therefore, $(0.1) = \frac{1}{2}(1.0) = \frac{1}{2}$, $(0.11) = \frac{1}{4}(11.0) = \frac{1}{4}(3) = \frac{3}{4}$ (etc)

Some expressions involving binary fractions

$$e^{2\pi i(0.0)} = 1, e^{2\pi i(0.1)} = -1$$

$$e^{2\pi i(1.0)} = 1, e^{2\pi i(1.1)} = -1$$

$$e^{2\pi i(0.01)} = i, e^{2\pi i(0.11)} = -i$$

Computing the QFT for $m = 2^n$ (3)

One way on seeing why this circuit works is to show:

1. For all $a_1 a_2 \dots a_n \in \{0,1\}^n$, on input state $|a_1 a_2 \dots a_n\rangle$ the output of the circuit (before reversing the qubits) is

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

2. And then

$$\begin{aligned} & (|0\rangle + e^{2\pi i(0.a_n)} |1\rangle) \dots (|0\rangle + e^{2\pi i(0.a_2 \dots a_n)} |1\rangle) (|0\rangle + e^{2\pi i(0.a_1 a_2 \dots a_n)} |1\rangle) \\ &= (|0\rangle + \omega^{2^{n-1}(a)} |1\rangle) \dots (|0\rangle + \omega^{2(a)} |1\rangle) (|0\rangle + \omega^{(a)} |1\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \omega^{(a)k} |k\rangle \quad (\text{where } \omega = e^{2\pi i/2^n}) \\ &= F_{2^n} |a_1 a_2 \dots a_n\rangle \end{aligned}$$

Exercise: show these two steps in detail

Summary of DLP algorithm

Implement $f(x) = g^{x_1} a^{-x_2} \pmod{p}$ reversibly
and F_{2^n} where $2^{n-1} < p - 1 < 2^n$



If the measured results are s_1 and s_2 where s_1 and $p - 1$
are coprime then output $r = -s_2/s_1 \pmod{p - 1}$
(otherwise, execute above circuit again)

Hidden Subgroup Problem framework

Aside: hidden subgroup problem (commutative version)

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 **coset** of H)

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

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

Example 2: $G = (\mathbb{Z}_{p-1})^2$ and

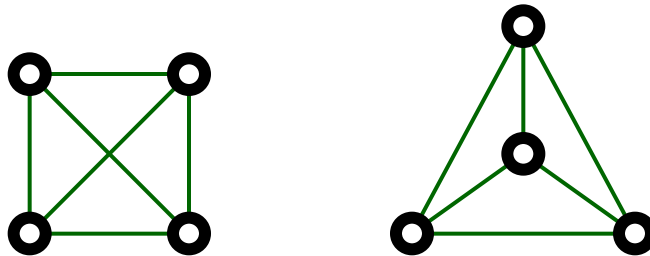
$H = \{(0,0), (r,1), (2r,2), \dots, ((p-2)r, p-2)\}$

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

Aside: hidden subgroup problem (noncommutative version)

Example 4: $G = S_n$ (the symmetric group, consisting of all permutations on n objects—which is not commutative) and H is any subgroup of G (and we use *left* cosets throughout)

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

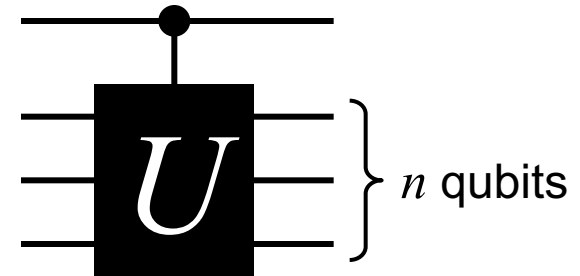
A simplified example

U is an unknown unitary operation on n qubits

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

Input: a black-box for a controlled- U

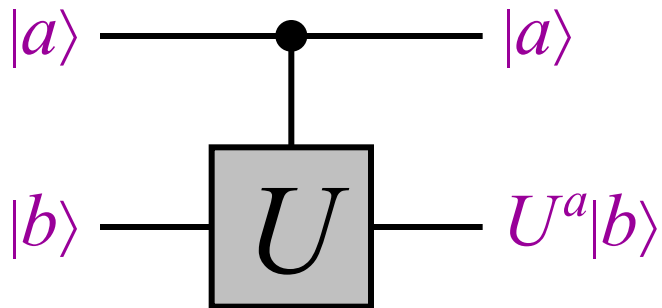
and a copy of the state $|\psi\rangle$



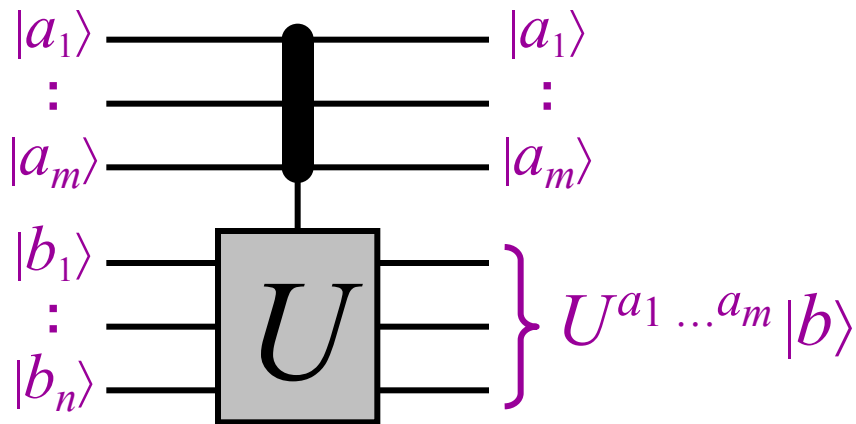
Output: the eigenvalue λ

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

Generalized controlled- U gates



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



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

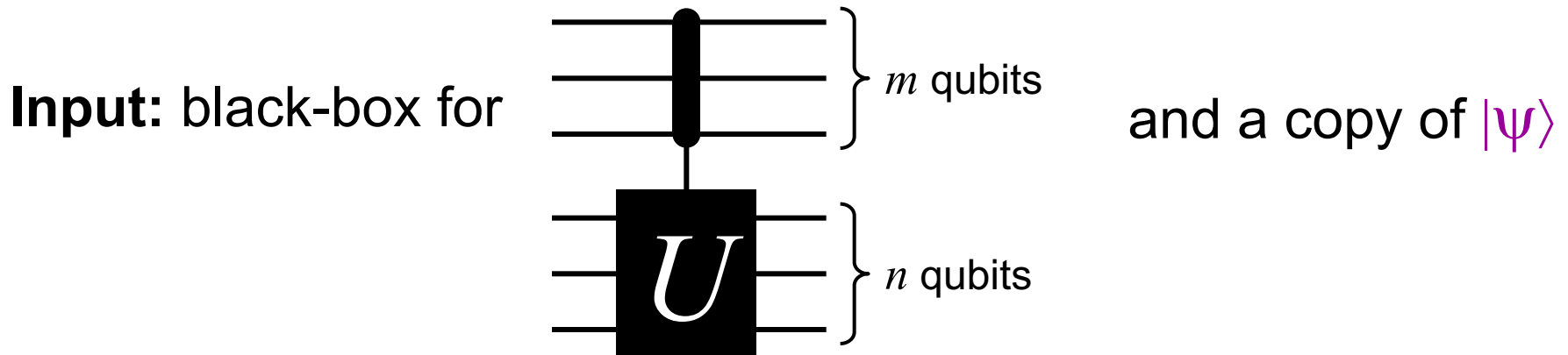
Example: $|1101\rangle|0101\rangle \mapsto |1101\rangle U^{1101} |0101\rangle$

Eigenvalue estimation problem

U is a unitary operation on n qubits

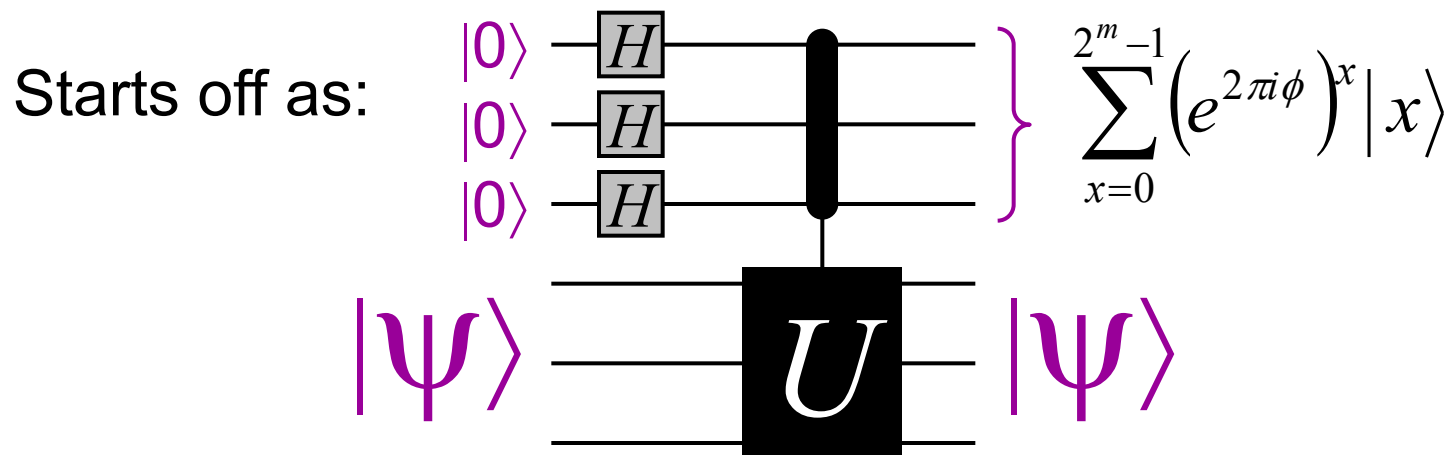
$|\psi\rangle$ is an eigenvector of U , with eigenvalue $e^{2\pi i\phi}$

($0 \leq \phi < 1$)



Output: ϕ (m -bit approximation)

Algorithm for eigenvalue estimation (1)



$$|00 \dots 0\rangle |\psi\rangle$$

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

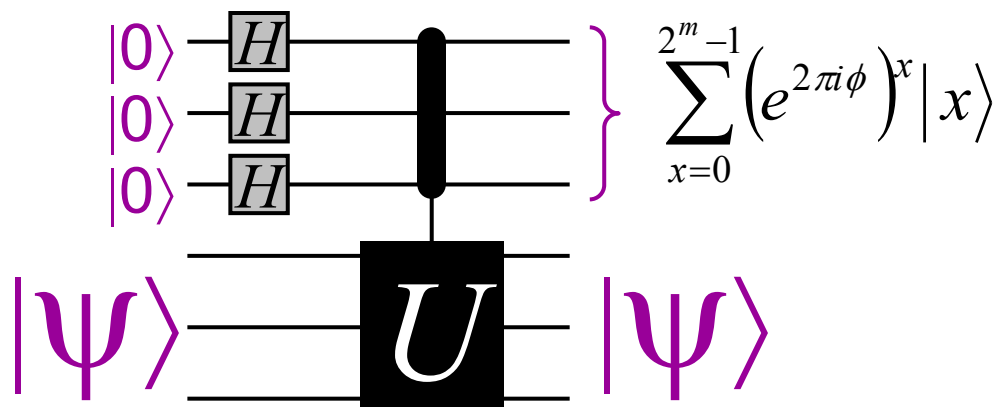
$$\mapsto (|0\rangle + |1\rangle)(|0\rangle + |1\rangle) \dots (|0\rangle + |1\rangle) |\psi\rangle$$

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

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

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

Algorithm for eigenvalue estimation (2)

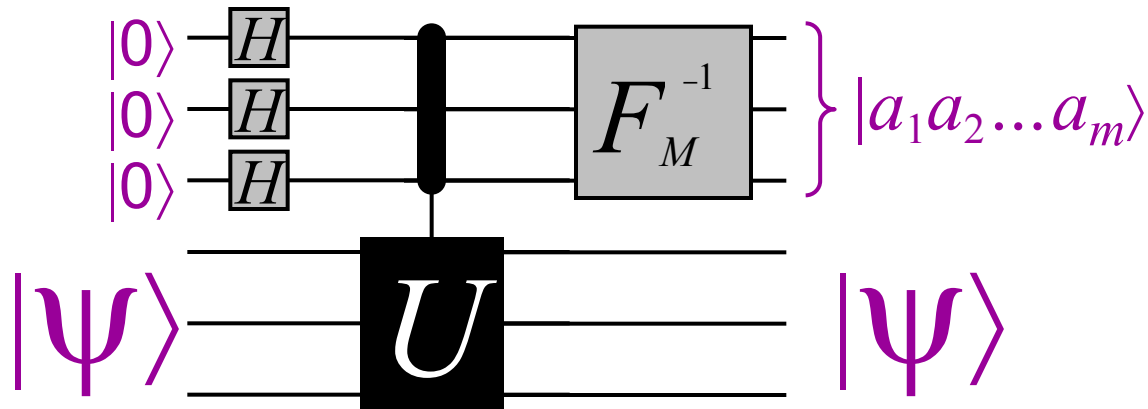


Recall that $F_M |a_1 a_2 \dots a_m\rangle = \sum_{x=0}^{2^m-1} (e^{2\pi i (0.a_1 a_2 \dots a_m)})^x |x\rangle$

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

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

Algorithm for eigenvalue estimation (3)



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

But what ϕ if is not of this nice form?

Example: $\phi = \frac{1}{3} = 0.0101010101010101\dots$

Algorithm for eigenvalue estimation (4)

What if ϕ is not of the nice form $\phi = 0.a_1a_2\dots a_m$?

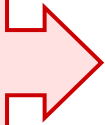
Example: $\phi = 1/3 = 0.\underline{0101010101010101}\dots$

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

Let $a/2^m = 0.a_1a_2\dots a_m$ be an m -bit approximation of ϕ ,
in the sense that $\phi = a/2^m + \delta$, where $|\delta| \leq 1/2^{m+1}$

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

What is the
amplitude of
 $|a_1a_2\dots a_m\rangle$?

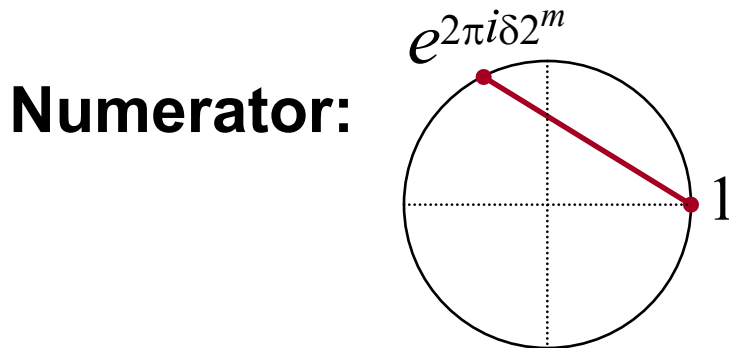


Algorithm for eigenvalue estimation (5)

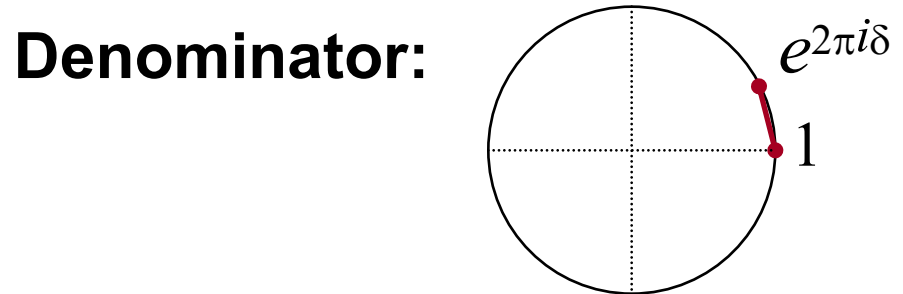
State is: $\frac{1}{2^m} \sum_{y=0}^{2^m-1} \sum_{x=0}^{2^m-1} e^{2\pi i(a-y)x/2^m} e^{2\pi i\delta x} |y\rangle$

geometric series!

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



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



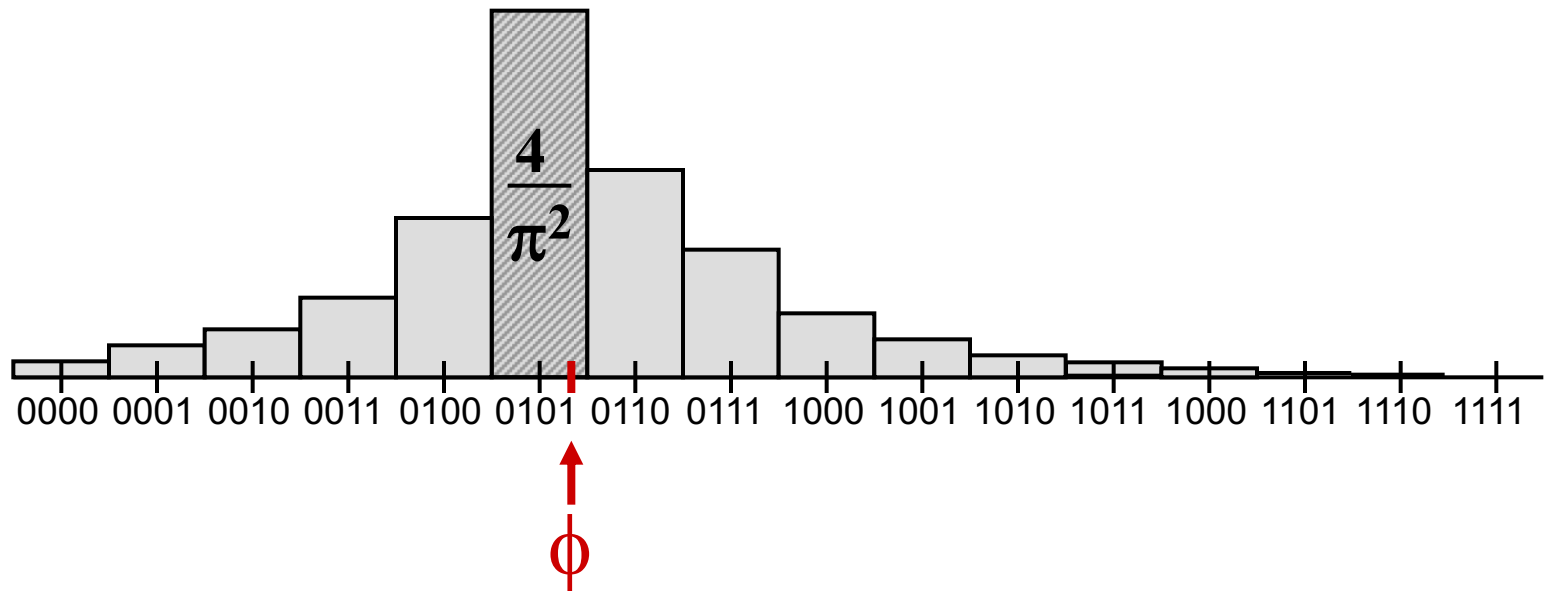
upper bounded by $2\pi\delta$

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 $\phi = 1/3 = 0.\underline{010101010101}01\dots$, the outcome probabilities look roughly like this:



Note: with $2m$ -qubit control gate, error probability is exponentially small 35