Introduction to Quantum Information Processing

CS 467 / CS 667
Phys 667 / Phys 767
C&O 481 / C&O 681

Lecture 7 (2008)

Richard Cleve
DC 2117
cleve@cs.uwaterloo.ca

Definition of the quantum Fourier transform (QFT)

Quantum Fourier transform

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

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

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

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

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!)

Quantum algorithm for DLP (1)

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 $f(x_1, x_2) = g^{x_1 - r x_2} \bmod p$

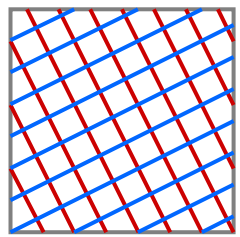
Thus, $f(x_1, x_2) = f(y_1, y_2)$ iff $x_1 - r x_2 \equiv y_1 - r y_2 \pmod{p-1}$

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: $f(x) = f(y)$ iff $x - y = kr \pmod 2$

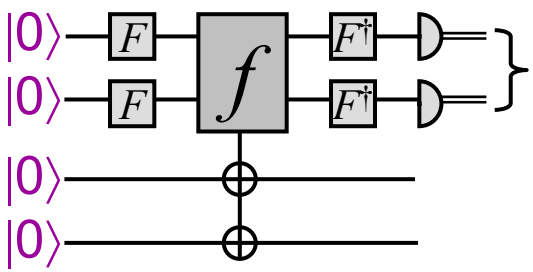


Quantum algorithm for DLP (2)

$f : \mathbf{Z}_{p-1} \times \mathbf{Z}_{p-1} \rightarrow \mathbf{Z}^*_p$ defined as $f(x_1, x_2) = g^{x_1} a^{-x_2} \bmod p$

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

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



result is a random (s, t) such that

$(s, t) \cdot (r, 1) \equiv 0 \pmod{p-1}$ **Why?**

if $\gcd(s, p-1) = 1$ then r can be computed as $r = -ts^{-1} \bmod p-1$ **Why?**

Details need to be filled in, but this is a sketch of how to efficiently solve DLP on a quantum computer

Introduction to Quantum Information Processing

CS 467 / CS 667
Phys 667 / Phys 767
C&O 481 / C&O 681

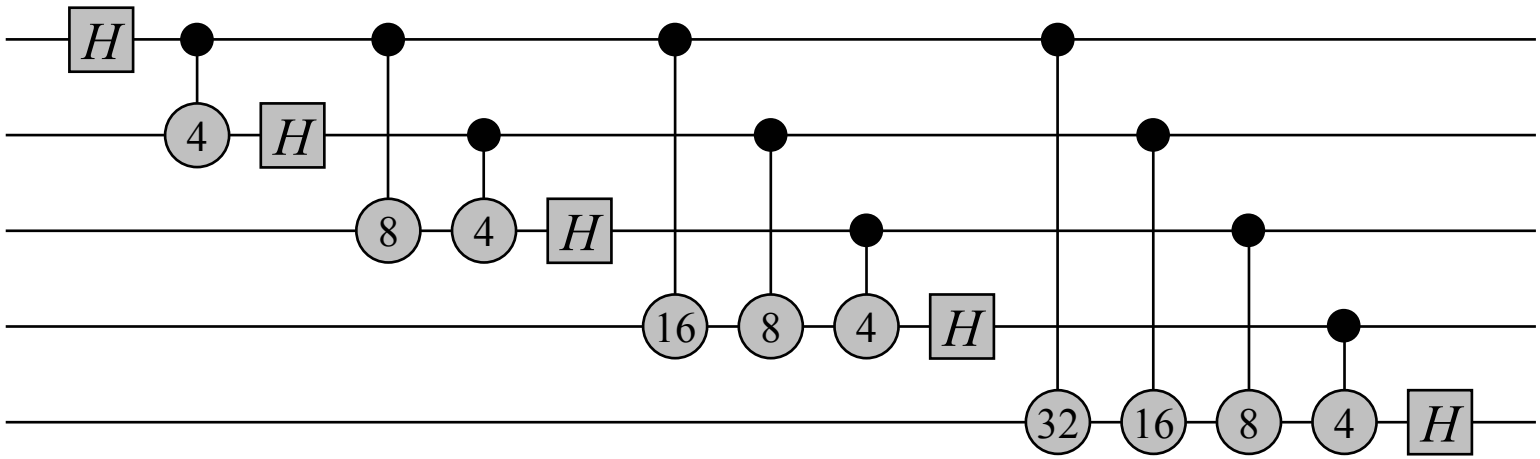
Lecture 8 (2008)

Richard Cleve
DC 2117
cleve@cs.uwaterloo.ca

Computing the QFT for $N = 2^n$

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

Quantum circuit for F_{32}:



reverse order

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

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

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

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

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

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

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

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

Contents

Loose ends in the discrete log algorithm

Quantum algorithm for DLP (3)

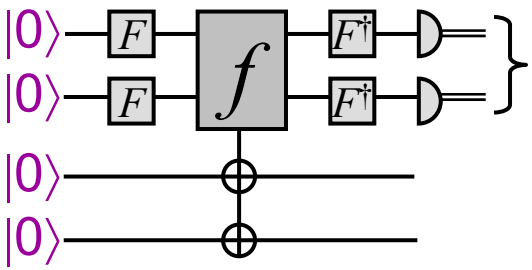
Input: p (n-bit prime), g (generator of $\mathbf{Z}^*_p$), $a \in \mathbf{Z}^*_p$

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 $\mathbf{Z}^*_7$)

Define $f : \mathbf{Z}_{p-1} \times \mathbf{Z}_{p-1} \rightarrow \mathbf{Z}^*_p$ as $f(x, y) = g^x a^{-y} \bmod p$

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

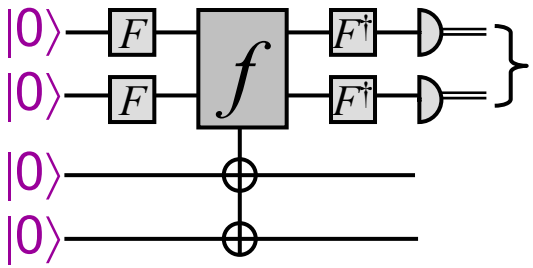


produces a random (s, t) such that

$(s, t) \cdot (r, 1) \equiv 0 \pmod{p-1}$

$\Leftrightarrow sr + t \equiv 0 \pmod{p-1}$

Quantum algorithm for DLP (4)



produces a random (s, t) such that

$$sr + t \equiv 0 \pmod{p-1}$$

If $\gcd(s, p-1) = 1$ then r can be computed as $r = -ts^{-1} \bmod p-1$

The probability that this occurs is $\phi(p-1)/(p-1)$, where ϕ is ***Euler's totient function***

It is known that $\phi(N) = \Omega(N/\log\log N)$, which implies that the above probability is at least $\Omega(1/\log\log p) = \Omega(1/\log n)$

Therefore, $O(\log n)$ repetitions are sufficient

… this is not bad—but things are actually better than that …

Quantum algorithm for DLP (5)

We obtain a random (s, t) such that $sr + t \equiv 0 \pmod{p-1}$

Note that each $s \in \{0, \ldots, p-2\}$ occurs with equal probability

Therefore, if we run the algorithm ***twice***: we obtain two independent samples $s_1, s_2 \in \{0, \ldots, p-2\}$

If it happens that $\gcd(s_1, s_2) = 1$ then (by Euclid) there exist integers a and b such that $as_1 + bs_2 = 1$ ➔ $r = -(at_1 + bt_2)$

Question: what is the probability that $\gcd(s_1, s_2) = 1$?

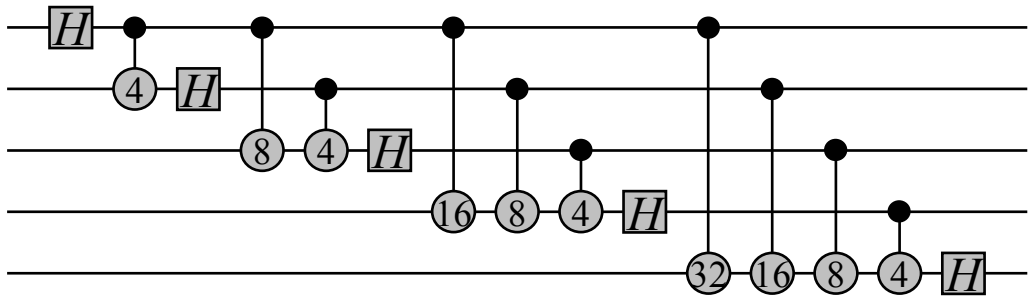
$$1 - \sum_{q \text{ prime}} \Pr[q \mid s_1] \Pr[q \mid s_2] > 1 - \sum_{q \text{ prime}} \frac{1}{q^2} > 0.54$$

Therefore, a ***constant*** number of repetitions suffices

Quantum algorithm for DLP (6)

***Another* loose end:** our algorithm uses QFTs modulo $p-1$, whereas we have only seen how to compute QFTs modulo 2^n

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



A variation of our QFT algorithm would work for moduli of the form 3^n, and, more generally, all ***smooth*** numbers (those that are products of "small" primes)

Quantum algorithm for DLP (7)

In fact, for the case where $p-1$ is smooth, there already exist polynomial-time ***classical*** algorithms for discrete log!

It's only the case where $p-1$ is ***not*** smooth that is interesting

Shor just used a modulus ***close to*** $p-1$, and, using careful error-analysis, showed that this was good enough ...

There are also ways of attaining good approximations of QFTs for arbitrary moduli—which we will see later on (so this loose end is not yet resolved)

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

The “hidden subgroup” 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 $H = \{0, r\}$

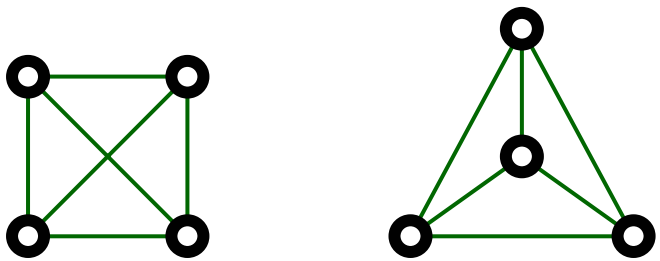
Example 2: $G = (\mathbf{Z}_{p-1})^2$ and
$H = \{(0,0), (r,1), (2r,2), \ldots, ((p-2)r, p-2)\}$

Example 3: $G = \mathbf{Z}$ and $H = r\mathbf{Z}$
(Shor's factoring algorithm can be viewed this way)

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 …



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

Introduction to Quantum Information Processing

CS 467 / CS 667
Phys 667 / Phys 767
C&O 481 / C&O 681

Lecture 9 (2008)

Richard Cleve
DC 2117
cleve@cs.uwaterloo.ca

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

Note: this is a major component of Shor's factoring algorithm

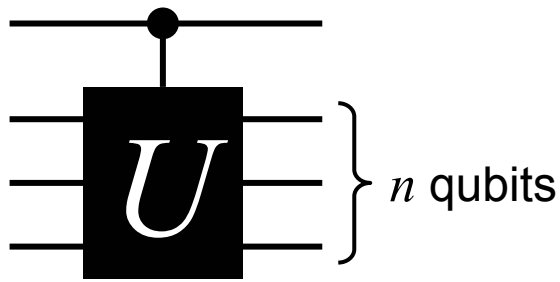
A simplified example

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

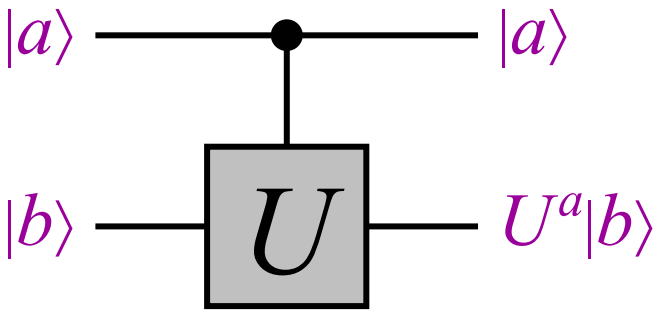
and a copy of the state $|\psi\rangle$



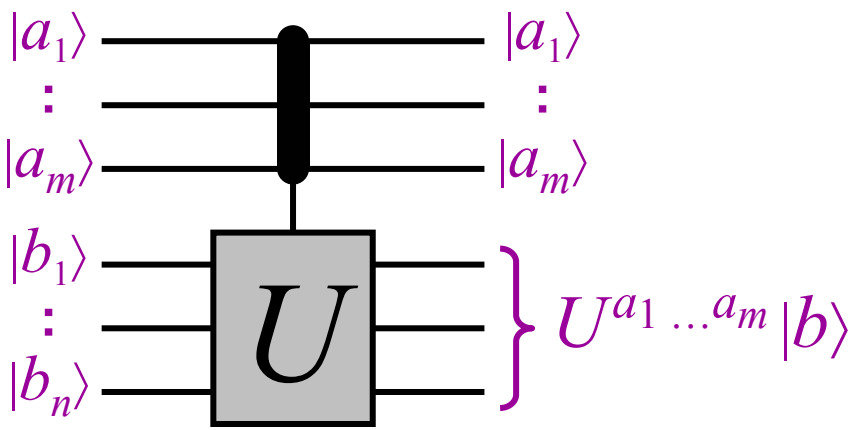
Output: the eigenvalue λ

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

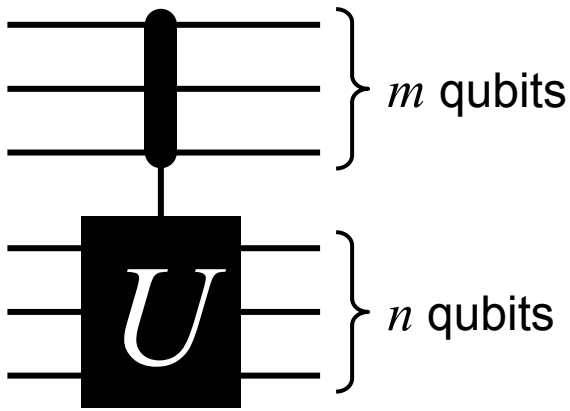
Eigenvalue estimation problem

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

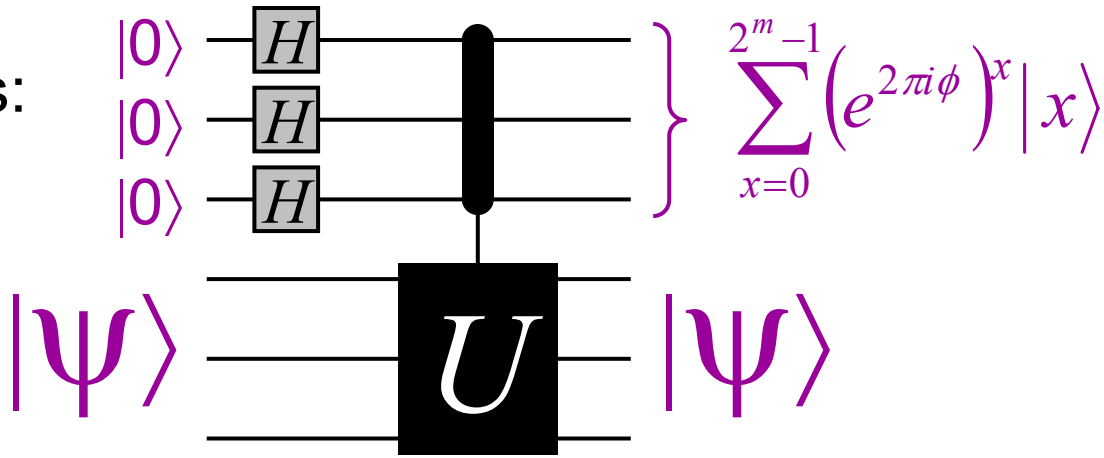


and a copy of $|\psi\rangle$

Output: ϕ (m-bit approximation)

Algorithm for eigenvalue estimation (1)

Starts off as:



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

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

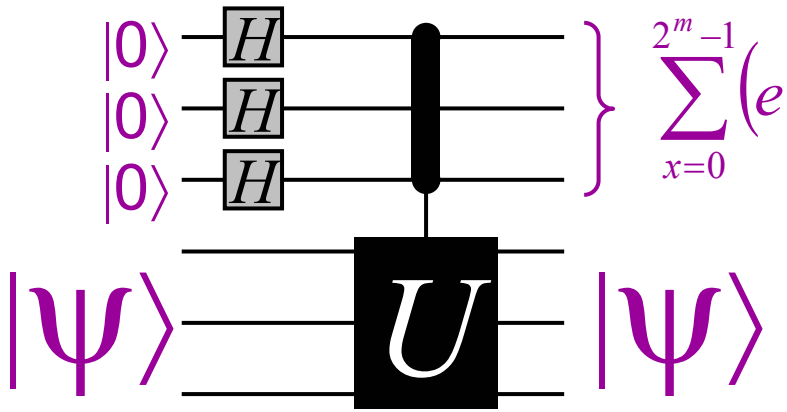
$\rightarrow (|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\ldots(|0\rangle + |1\rangle)|\psi\rangle$

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

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

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

Algorithm for eigenvalue estimation (2)



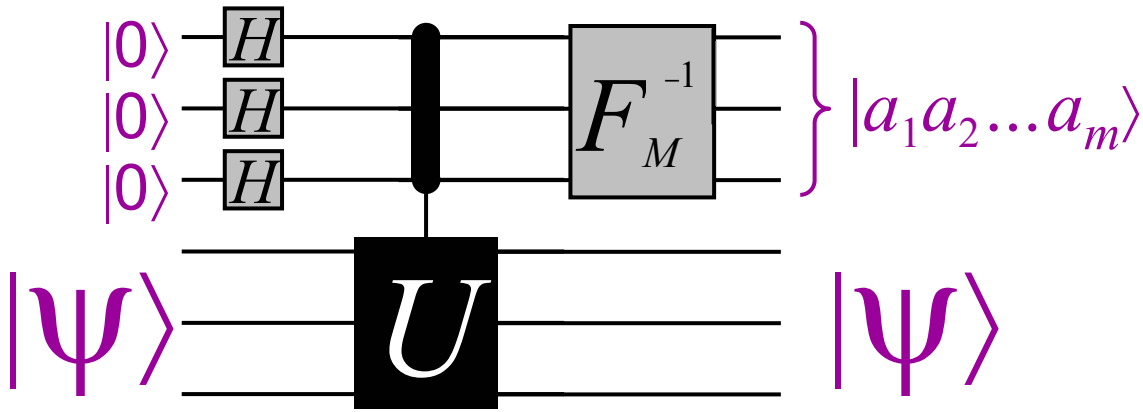
$$\sum_{x=0}^{2^m-1} \left(e^{2\pi i\phi}\right)^x |x\rangle$$

When $\phi = 0.a_1a_2\ldots a_m$: $F_M |a_1a_2\ldots a_m\rangle = \sum_{x=0}^{2^m-1} \left(e^{2\pi i\phi}\right)^x |x\rangle$

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

Therefore, applying the ***inverse*** of F_M yields the digits of ϕ

Algorithm for eigenvalue estimation (3)



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

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: $\phi = \frac{1}{3} = 0.\underline{01}01010101010101\ldots$

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 $|\delta| \leq 1/2^{m+1}$

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

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

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

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

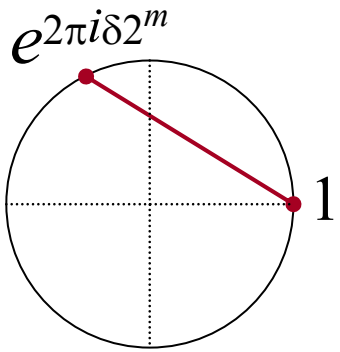
Algorithm for eigenvalue estimation (5)

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!

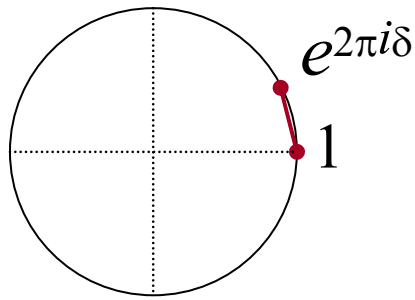
The amplitude of $|y\rangle$, for $y = a$ is $\frac{1}{2^m}\sum_{x=0}^{2^m-1} e^{2\pi i\delta x} = \frac{1}{2^m}\frac{1-\left(e^{2\pi i\delta}\right)^{2^m}}{1-e^{2\pi i\delta}}$

Numerator:



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

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 $\phi = ⅓ = 0.\underline{0101}0101010101\ldots$, the outcome probabilities look roughly like this: