Title: PSI 2016/2017 Quantum Information (Review) - Lecture 11 (Michele Mosca)

Date: Mar 07, 2017 02:00 PM

URL: http://pirsa.org/17030035

Abstract:

PERIMETER INSTITUTE FOR THEORETICAL PHYSICS

Introduction to Quantum Algorithms

Michele Mosca

7 March 2017

UNIVERSITY OF WATERLOO | IQC Institute for Quantum Computing

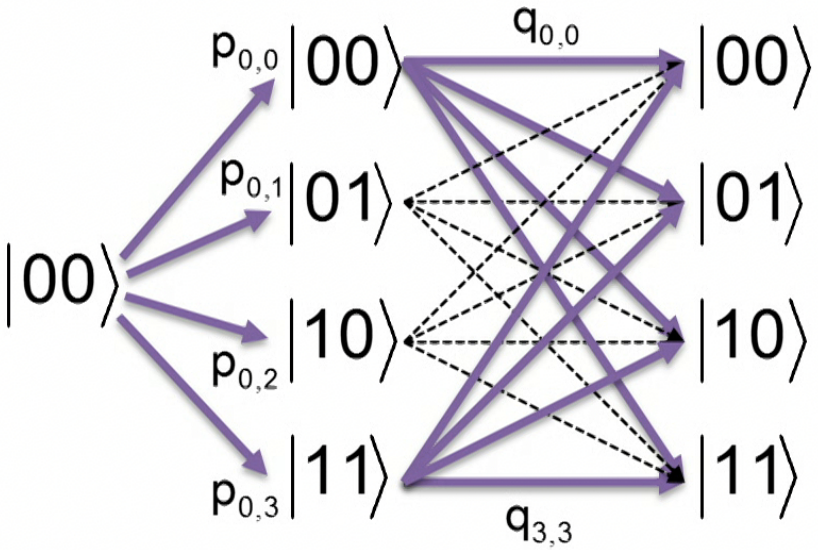
CryptoWorks21

evolution

The art of quantum algorithmics is to choreograph *constructive interference* on *desirable outcomes* and *destructive interference* on *undesirable outcomes*.

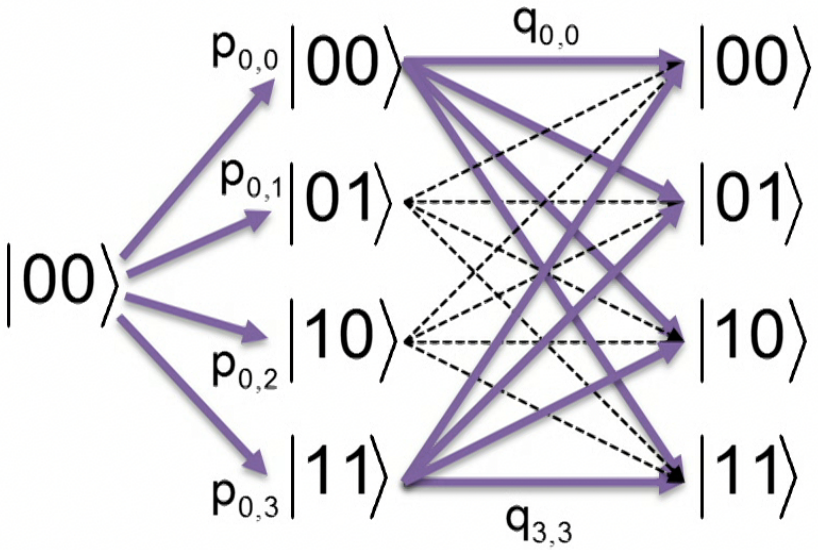
A classical randomized algorithm

- Several computational paths leading to the same outcome.
- Add up the probabilities.



A classical randomized algorithm

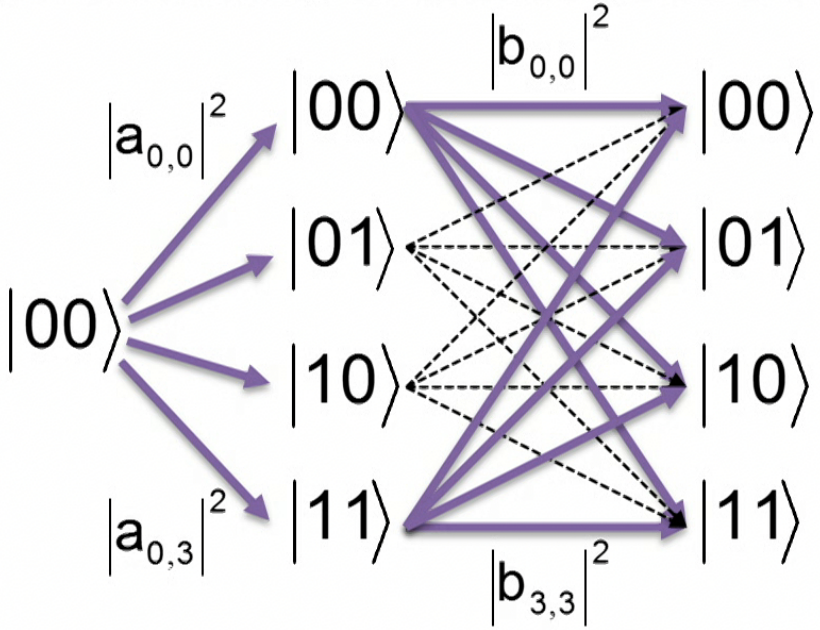
- Several computational paths leading to the same outcome.
- Add up the probabilities.



$$\Pr(00) = \sum_j p_{0,j} q_{j,0}$$

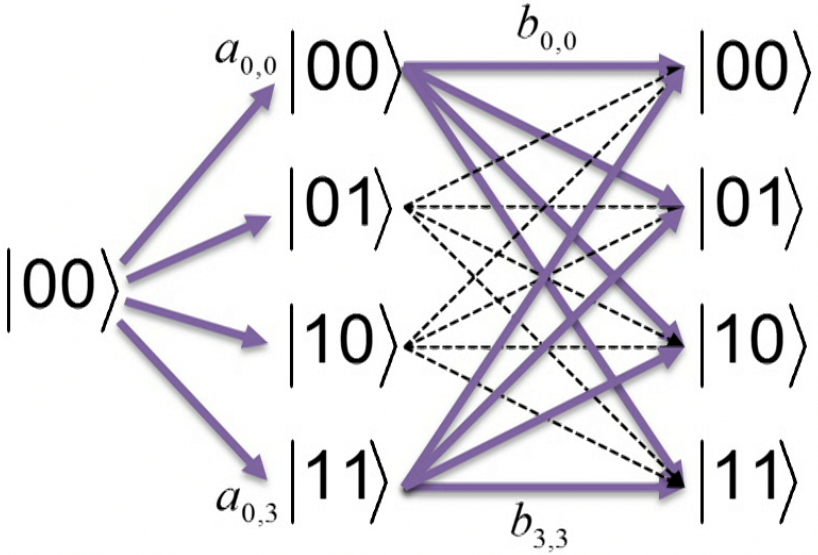
A classical randomized algorithm

The probabilities could correspond to the square of a probability amplitude (due to measuring the quantum system at each time step)



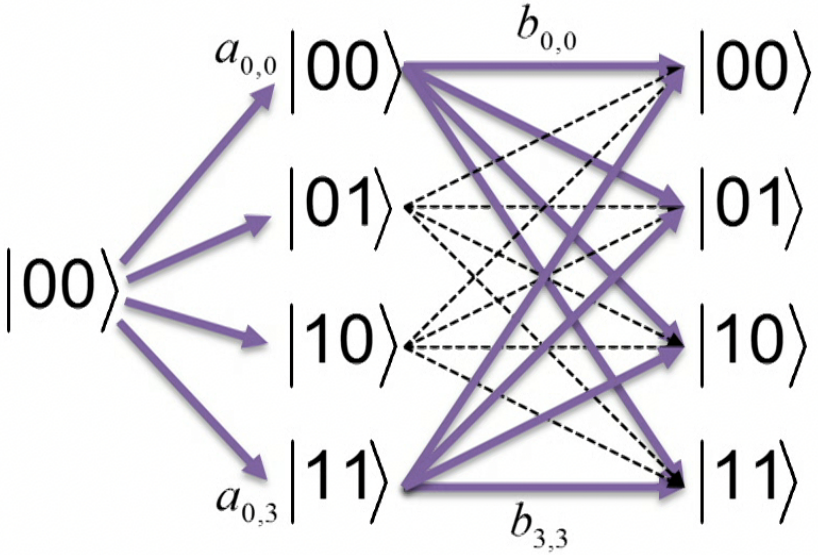
A quantum algorithm

If we don't measure at each time step, only at the end, the probability amplitudes first have a chance to interfere.



A quantum algorithm

If we don't measure at each time step, only at the end, the probability amplitudes first have a chance to interfere.



$$\Pr(00) = \left| \sum_j a_{0,j} b_{j,0} \right|^2$$

How do quantum algorithms work?

Given a polynomial-time classical algorithm for $f:\{0,1\}^n \longrightarrow T$, it is straightforward to construct a quantum algorithm that creates the state

$$\frac{1}{\sqrt{2^n}}\sum_x |x\rangle|f(x)\rangle$$

at the cost of about ***one*** evaluation of f

Is this exponentially many computations at polynomial cost?

No! — the most straightforward way of extracting information from the state yields just $(x, f(x))$ for a random $x \in \{0,1\}^n$

How do quantum algorithms work?

Given a polynomial-time classical algorithm for $f:\{0,1\}^n \longrightarrow T$, it is straightforward to construct a quantum algorithm that creates the state

$$\frac{1}{\sqrt{2^n}}\sum_x |x\rangle|f(x)\rangle$$

at the cost of about ***one*** evaluation of f

Is this exponentially many computations at polynomial cost?

No! — the most straightforward way of extracting information from the state yields just $(x, f(x))$ for a random $x \in \{0,1\}^n$

But we can make some interesting ***tradeoffs***:

instead of learning about any $(x, f(x))$ point, one can learn something about a ***global property*** of f

Quantum algorithms

- Quantum Algorithms should exploit quantum parallelism *and* quantum interference.

- This is necessary, but not sufficient, in order to outperform a classical probabilistic algorithm. E.g. a pure state description of the algorithm should have a substantial amount of entanglement.

Query scenario

Input: a function f, given as a black box (a.k.a. oracle)

Goal: determine some information about f making as few queries to f as possible (of course, other operations are allowed – but we do not count them)

Example: polynomial interpolation

Let: $f(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_d x^d$

Goal: determine $c_0, c_1, c_2, \ldots, c_d$

Question: How many f-queries does one require for this?

Query scenario

Input: a function f, given as a black box (a.k.a. oracle)

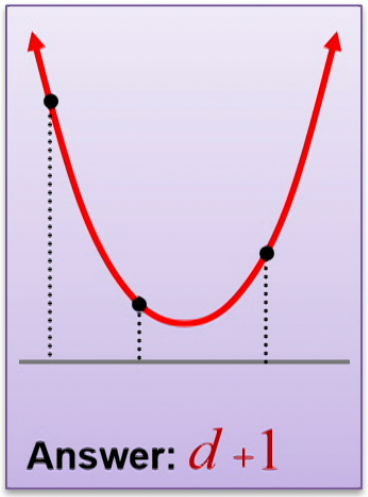
Goal: determine some information about f making as few queries to f as possible (of course, other operations are allowed – but we do not count them)

Example: polynomial interpolation

Let: $f(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_d x^d$

Goal: determine $c_0, c_1, c_2, \ldots, c_d$

Question: How many f-queries does one require for this?



Deutsch's problem

Let $f: \{0,1\} \rightarrow \{0,1\}$

There are ***four*** possibilities:

x	$f_1(x)$
0	0
1	0

x	$f_2(x)$
0	1
1	1

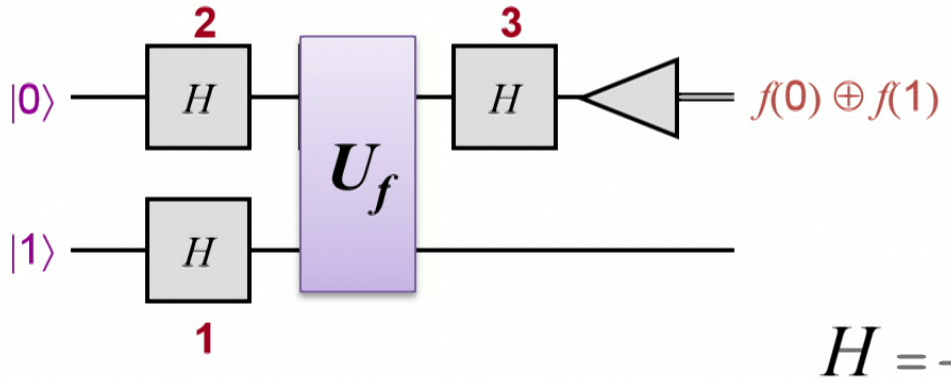
x	$f_3(x)$
0	0
1	1

x	$f_4(x)$
0	1
1	0

Goal: determine whether or not $f(0) = f(1)$ (i.e. $f(0) \oplus f(1)$)

Any classical method requires ***two*** queries

Quantum algorithm for Deutsch



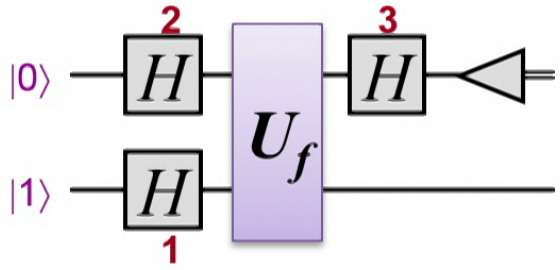
1 query + **4** auxiliary operations

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

How does this algorithm work?

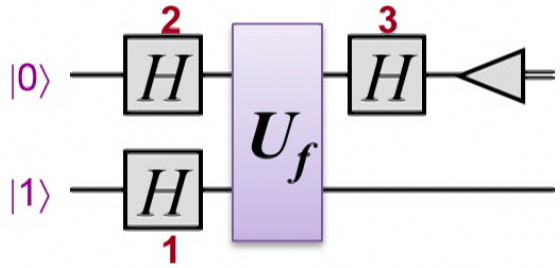
Each of the three H operations can be seen as playing a different role ...

Quantum algorithm (1)



12

Quantum algorithm (1)

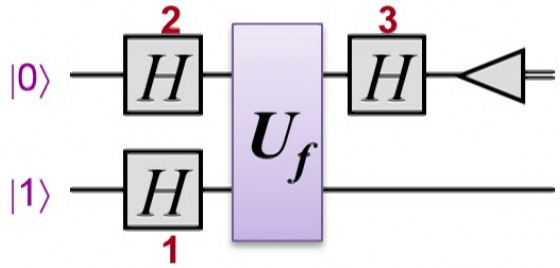


1. Creates the state $|0\rangle - |1\rangle$, which is an eigenvector of

$$\begin{cases} \textbf{NOT} \text{ with eigenvalue } -1 \\ I \quad \text{with eigenvalue } +1 \end{cases}$$

U_f

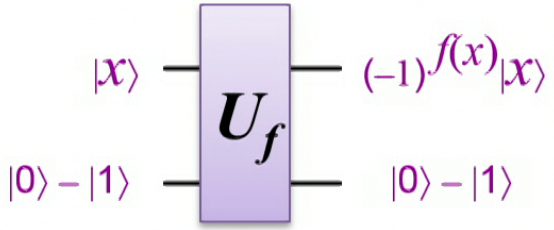
Quantum algorithm (1)

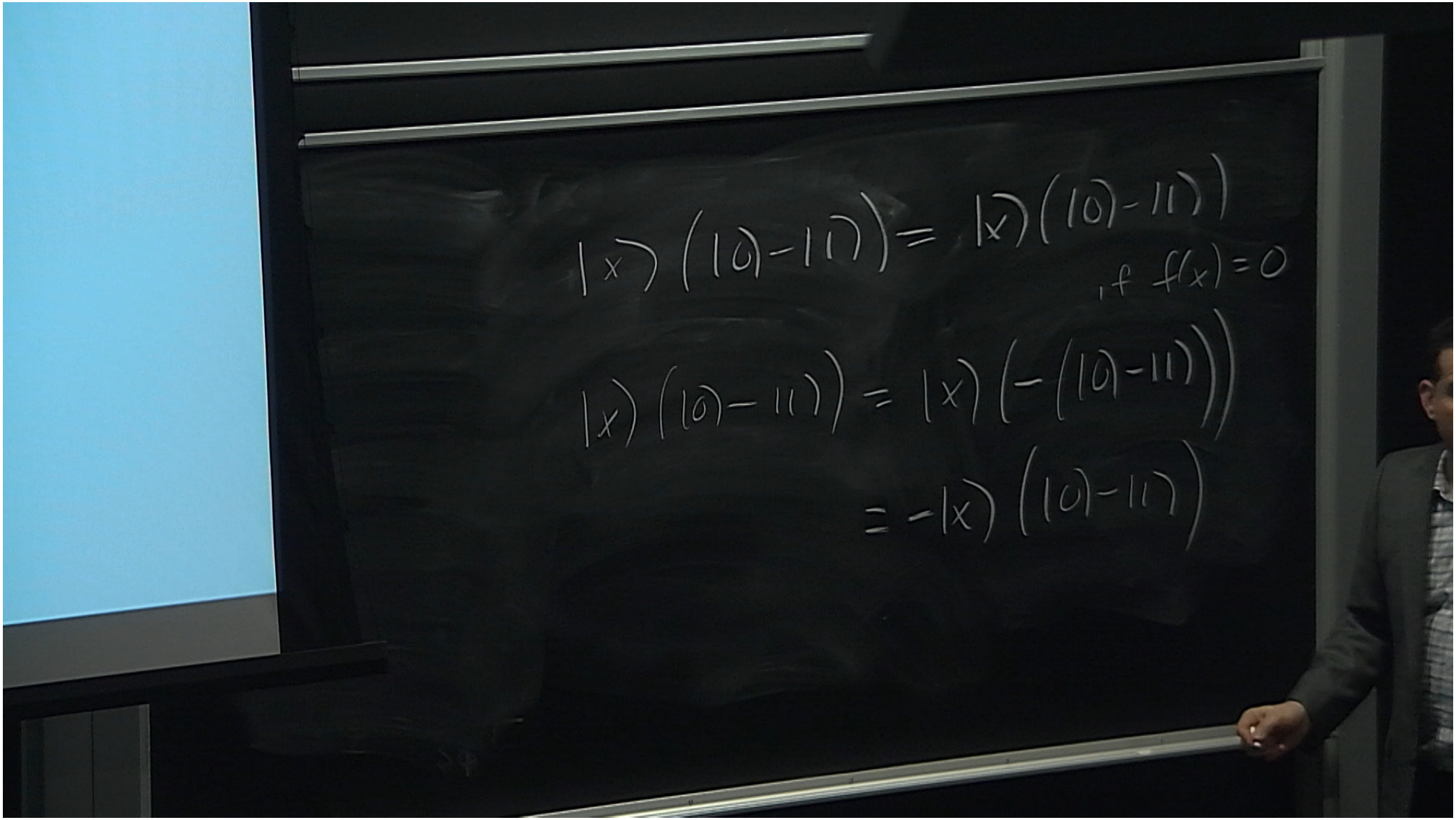


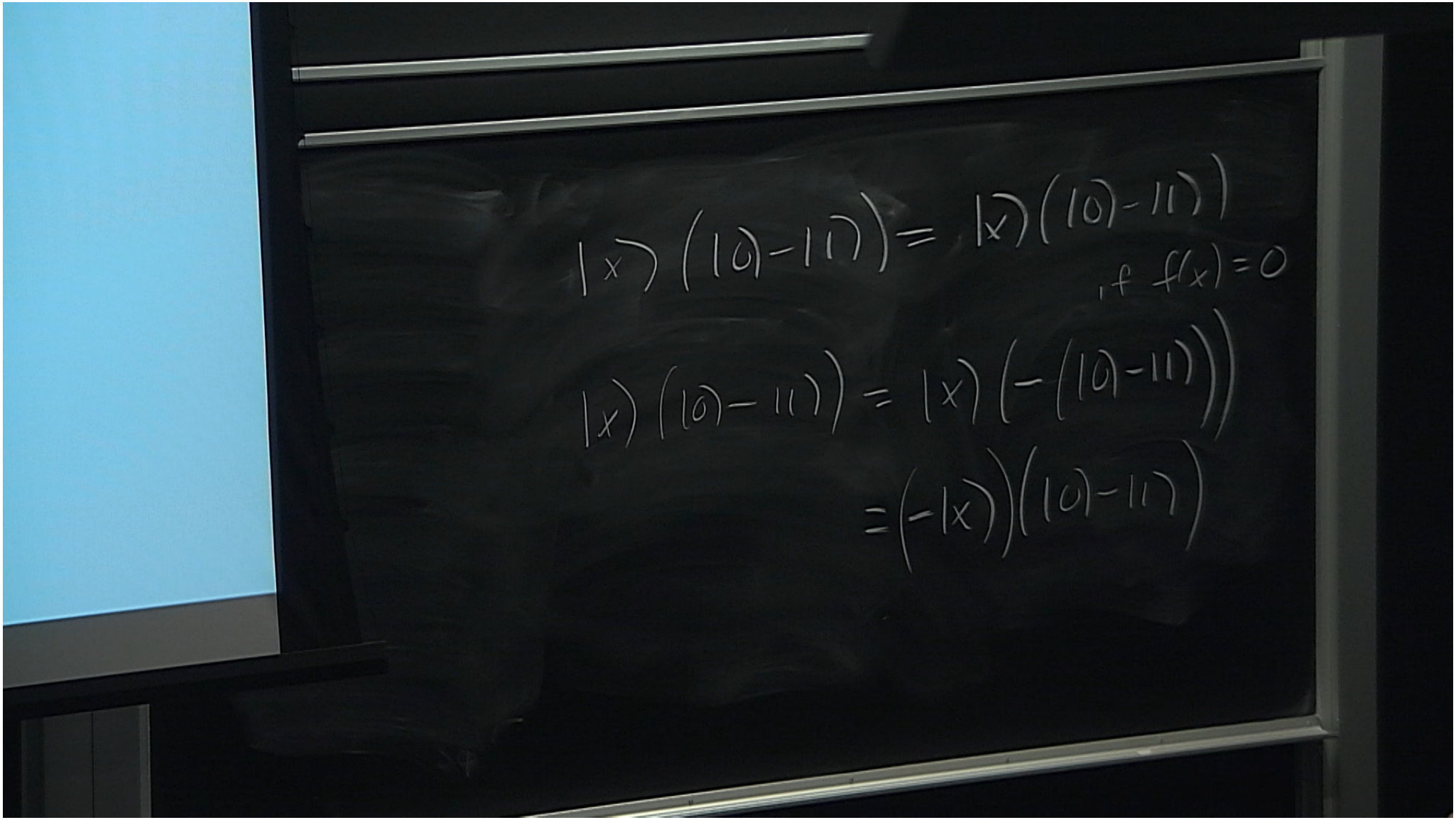
1. Creates the state $|0\rangle - |1\rangle$, which is an eigenvector of

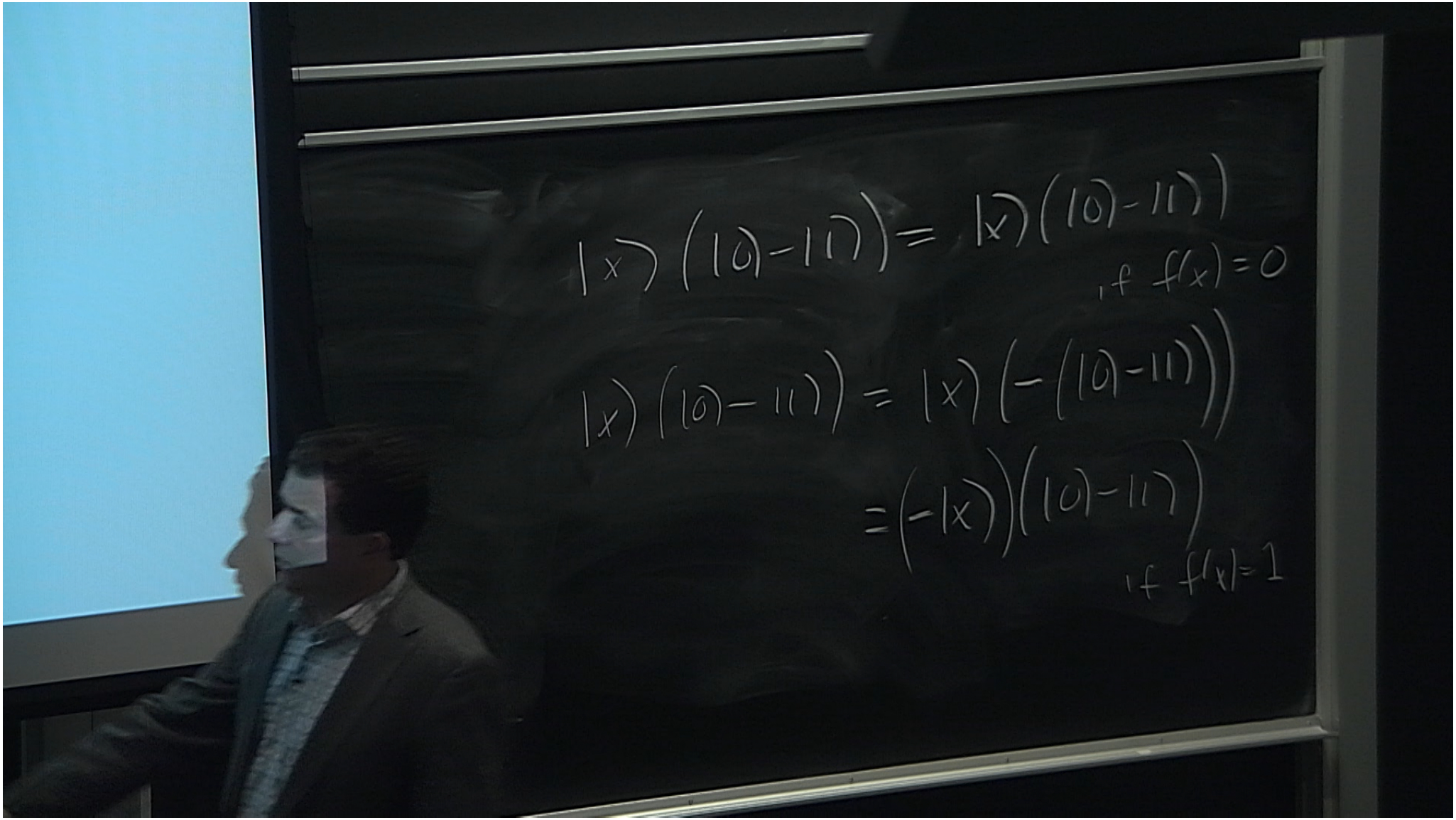
$$\begin{cases} \textbf{NOT} \text{ with eigenvalue } -1 \\ I \quad \text{with eigenvalue } +1 \end{cases}$$

This causes f to induce a ***phase shift*** of $(-1)^{f(x)}$ to $|x\rangle$

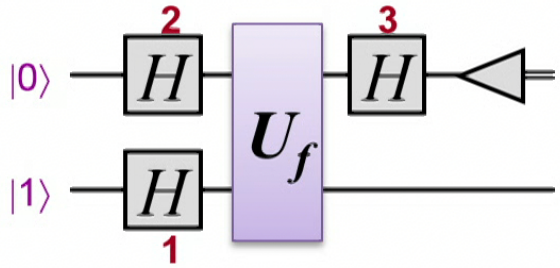



|x⟩(|0⟩-|1⟩) = |x⟩(|0⟩-|1⟩)
if f(x)=0
|x⟩(|0⟩-|1⟩) = |x⟩(-(|0⟩-|1⟩))
= -|x⟩(|0⟩-|1⟩)


|x⟩(|0⟩-|1⟩) = |x⟩(|0⟩-|1⟩)
if f(x)=0
|x⟩(|0⟩-|1⟩) = |x⟩(-(|0⟩-|1⟩))
=(-|x⟩)(|0⟩-|1⟩)


$|x\rangle(|0\rangle-|1\rangle) = |x\rangle(|0\rangle-|1\rangle)$
if $f(x)=0$
$|x\rangle(|0\rangle-|1\rangle) = |x\rangle(-(|0\rangle-|1\rangle))$
$=(-|x\rangle)(|0\rangle-|1\rangle)$
if $f(x)=1$

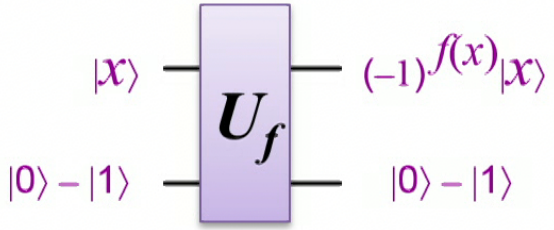
Quantum algorithm (1)



1. Creates the state $|0\rangle - |1\rangle$, which is an eigenvector of

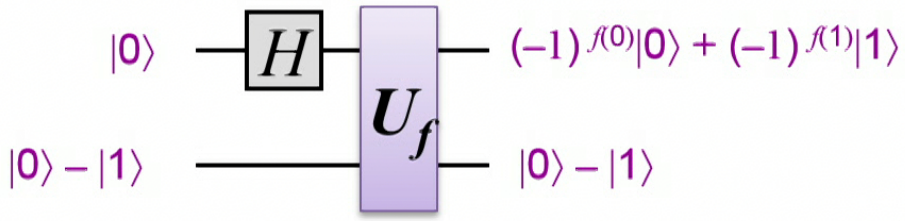
$$\begin{cases} \textbf{NOT} \text{ with eigenvalue } -1 \\ I \quad \text{with eigenvalue } +1 \end{cases}$$

This causes f to induce a ***phase shift*** of $(-1)^{f(x)}$ to $|x\rangle$



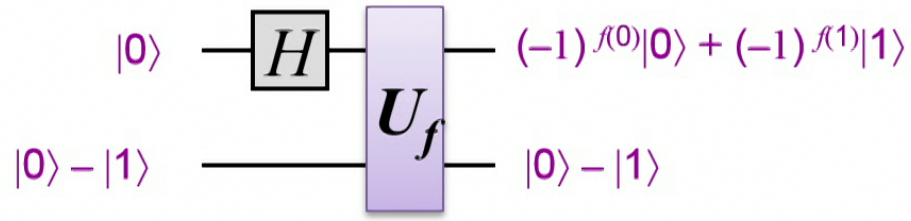
Quantum algorithm (2)

2. Causes f to be queried ***in superposition*** (at $|0\rangle + |1\rangle$)



Quantum algorithm (2)

2. Causes f to be queried ***in superposition*** (at $|0\rangle + |1\rangle$)



x	$f_1(x)$
0	0
1	0

x	$f_2(x)$
0	1
1	1

$\pm(|0\rangle + |1\rangle)$

x	$f_3(x)$
0	0
1	1

x	$f_4(x)$
0	1
1	0

$\pm(|0\rangle - |1\rangle)$

Quantum algorithm (3)

3. Distinguishes between $\pm(|0\rangle + |1\rangle)$ and $\pm(|0\rangle - |1\rangle)$

$$\pm(|0\rangle + |1\rangle) \xleftrightarrow{H} \pm|0\rangle$$

$$\pm(|0\rangle - |1\rangle) \xleftrightarrow{H} \pm|1\rangle$$

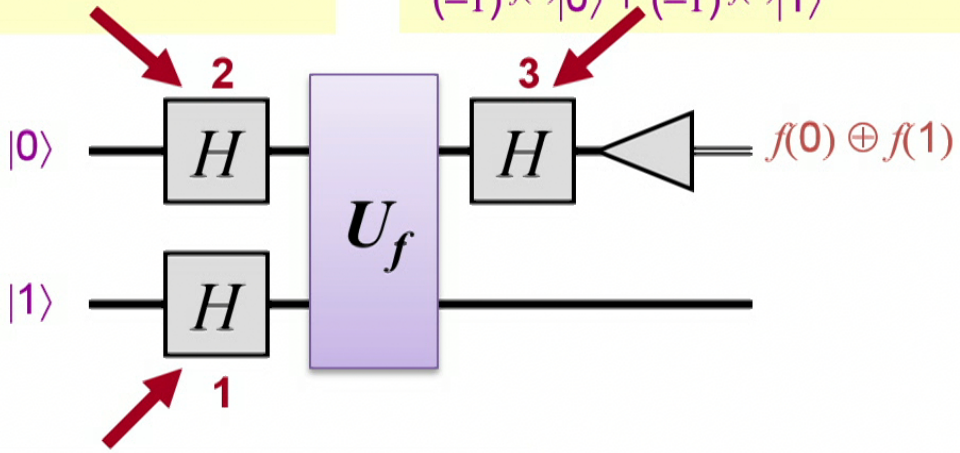
Summary of Deutsch's algorithm

Makes only one query, whereas two are needed classically

produces superpositions of inputs to f: $|0\rangle + |1\rangle$

extracts phase differences from

$(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle$



constructs eigenvector so f-queries induce phases: $|x\rangle \rightarrow (-1)^{f(x)}|x\rangle$

Aside: This was the first quantum algorithm implemented (1997)

JOURNAL OF CHEMICAL PHYSICS VOLUME 109, NUMBER 5 1 AUGUST 1998

Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer

J. A. Jones[a)]
Oxford Centre for Molecular Sciences, New Chemistry Laboratory, South Parks Road, Oxford OX1 3QT, United Kingdom and Centre for Quantum Computation, Clarendon Laboratory, Parks Road, Oxford OX1 3PU, United Kingdom

M. Mosca
Centre for Quantum Computation, Clarendon Laboratory, Parks Road, Oxford OX1 3PU, United Kingdom and Mathematical Institute, 24-29 St Giles', Oxford, OX1 3LB, United Kingdom

(Received 16 January 1998; accepted 22 April 1998)

Quantum computing shows great promise for the solution of many difficult problems, such as the simulation of quantum systems and the factorization of large numbers. While the theory of quantum computing is fairly well understood, it has proved difficult to implement quantum computers in real physical systems. It has recently been shown that nuclear magnetic resonance (NMR) can be used to implement small quantum computers using the spin states of nuclei in carefully chosen small molecules. Here we demonstrate the use of a NMR quantum computer based on the pyrimidine base cytosine, and the implementation of a quantum algorithm to solve Deutsch's problem (distinguishing between constant and balanced functions). This is the first successful implementation of a quantum algorithm on any physical system. © *1998 American Institute of Physics*. [S0021-9606(98)00729-6]

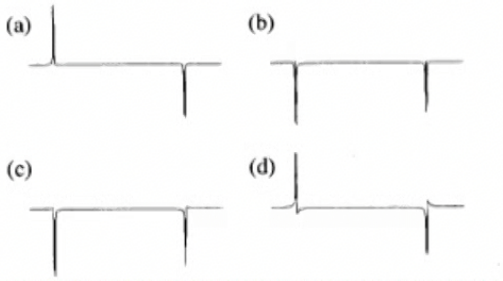
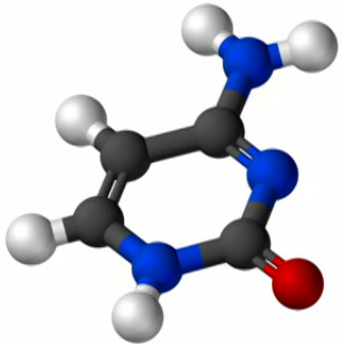


FIG. 6. Experimental implementation of a quantum algorithm to determine $f(0) \oplus f(1)$ on a NMR quantum computer. In this case the result can be read out on spin I, that is, using the signals on the left of the spectrum. For details of the labeling, see Fig. 4. As expected, the I spin is inverted when the function is balanced (f_{01} or f_{10}), but not when the function is constant (f_{00} or f_{11}).



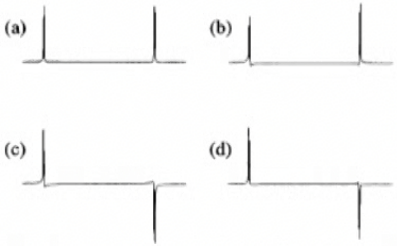


FIG. 4. Experimental implementation of an algorithm to determine $f(0)$ on a NMR quantum computer. (a) The result of applying U_{f00}; as this propagator is the identity matrix this spectrum can also serve as a reference. The left-hand pair of signals corresponds to the first spin (I), while the pair on the right-hand side correspond to the second spin (S). Note that the signals

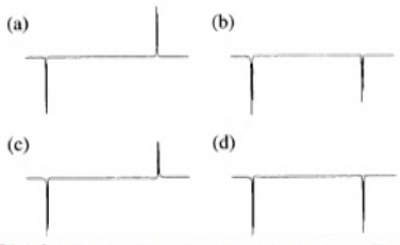


FIG. 5. Experimental implementation of an algorithm to determine $f(1)$ on a NMR quantum computer; in this case the algorithm starts with spin I in the excited state, $|1\rangle$, and so signals from spin I are in emission. For details of the labeling see Fig. 4.

Constant vs. balanced

Let $f: \{0,1\}^n \rightarrow \{0,1\}$ be either constant or balanced, where

- ***constant*** means $f(x) = 0$ for all x, or $f(x) = 1$ for all x
- ***balanced*** means $\Sigma_x \ f(x) = \ 2^{n-1}$

Constant vs. balanced

Let $f: \{0,1\}^n \rightarrow \{0,1\}$ be either constant or balanced, where

- ***constant*** means $f(x) = 0$ for all x, or $f(x) = 1$ for all x
- ***balanced*** means $\Sigma_x \ f(x) = \ 2^{n-1}$

Goal: determine whether f is constant or balanced

How many queries are there needed *classically?* $2^{n-1}+1$

Example: if $f(0000) = f(0001) = f(0010) = \ldots = f(0111) = 0$ then it still could be either

Constant vs. balanced

Let $f: \{0,1\}^n \rightarrow \{0,1\}$ be either constant or balanced, where

- ***constant*** means $f(x) = 0$ for all x, or $f(x) = 1$ for all x
- ***balanced*** means $\Sigma_x \ f(x) = \ 2^{n-1}$

Goal: determine whether f is constant or balanced

How many queries are there needed *classically?* $2^{n-1}+1$

Example: if $f(0000) = f(0001) = f(0010) = \ldots = f(0111) = 0$ then it still could be either

Quantumly?

Constant vs. balanced

Let $f: \{0,1\}^n \rightarrow \{0,1\}$ be either constant or balanced, where

- ***constant*** means $f(x) = 0$ for all x, or $f(x) = 1$ for all x
- ***balanced*** means $\Sigma_x \ f(x) = \ 2^{n-1}$

Goal: determine whether f is constant or balanced

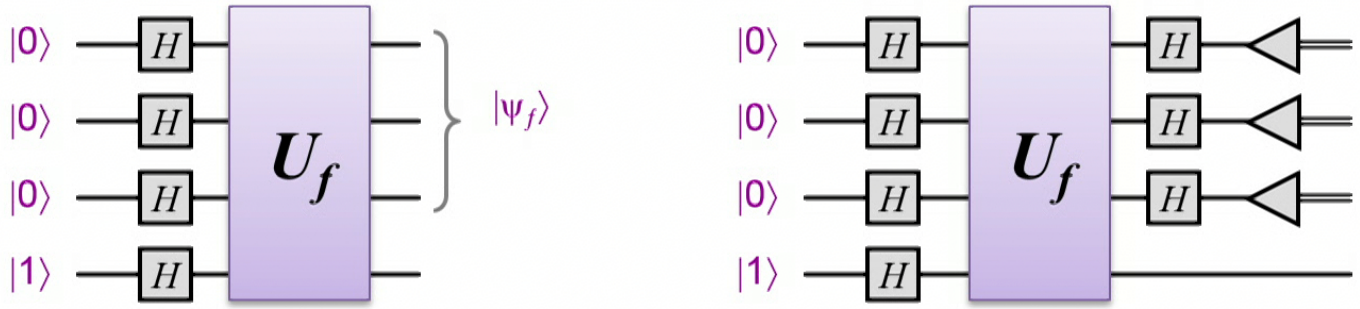
How many queries are there needed *classically?* $2^{n-1}+1$

Example: if $f(0000) = f(0001) = f(0010) = \ldots = f(0111) = 0$ then it still could be either

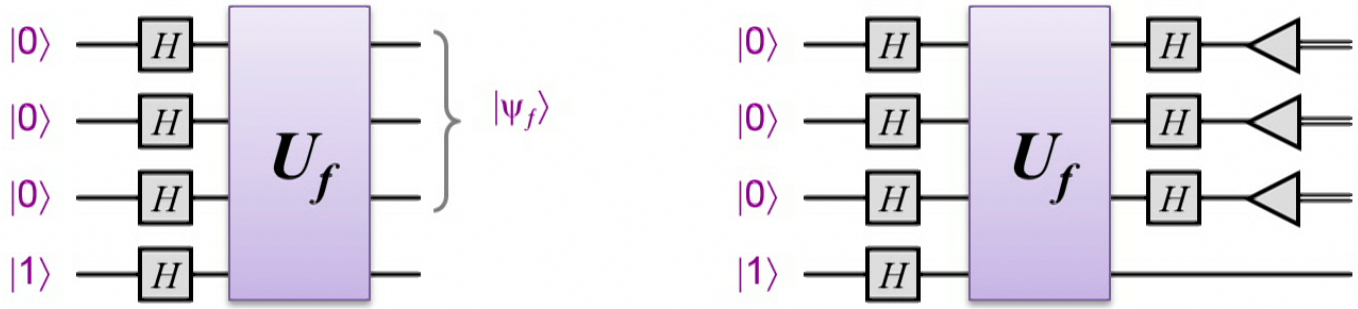
Quantumly? just 1 query suffices!

[Deutsch & Jozsa, 1992]

Quantum algorithm



Quantum algorithm



Constant case: $|\psi_f\rangle \equiv |\psi_{\text{const}}\rangle = \pm \sum_x |x\rangle$ ***Why?***

Balanced case: $|\psi_f\rangle \equiv |\psi_{\text{bal}}\rangle$ is ***orthogonal*** to $\pm \sum_x |x\rangle$ ***Why?***

How to distinguish between the cases? What is $H^{\otimes n}|\psi_f\rangle$?

Constant case: $H^{\otimes n}|\psi_{\text{const}}\rangle = \pm |00\ldots0\rangle$

Balanced case: $H^{\otimes n}|\psi_{\text{bal}}\rangle$ is orthogonal to $H^{\otimes n}|\psi_{\text{const}}\rangle = \pm |0\ldots00\rangle$, by unitarity of $H^{\otimes n}$

Last step of the algorithm: if the measured result is 000 then output "constant", otherwise output "balanced"

Quantum vs. classical separations

Black-box problem	Quantum	Classical
Deutsch's problem	**1** (query)	**2** (queries)
constant vs. balanced	**1**	$\frac{1}{2}\,\mathbf{2}^{n} + \mathbf{1}$

(only for exact)

Quantum vs. classical separations

Black-box problem	Quantum	Classical	
Deutsch's problem	1 (query)	2 (queries)	
constant vs. balanced	1	$\frac{1}{2}\,2^n + 1$	(only for exact)
Simon's problem	$O(n)$	$\theta(2^{n/2})$	(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 $x = y$

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?

Classical lower bound

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

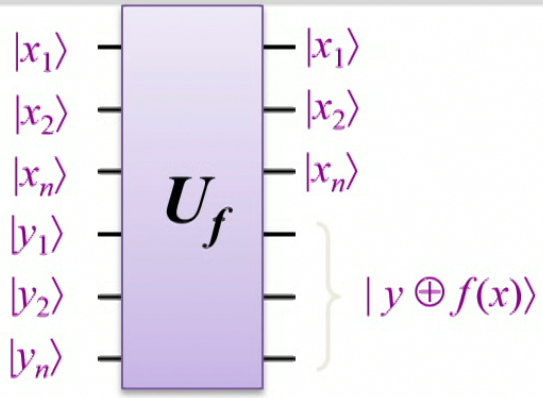
Proof is omitted here

Note: 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 (1)

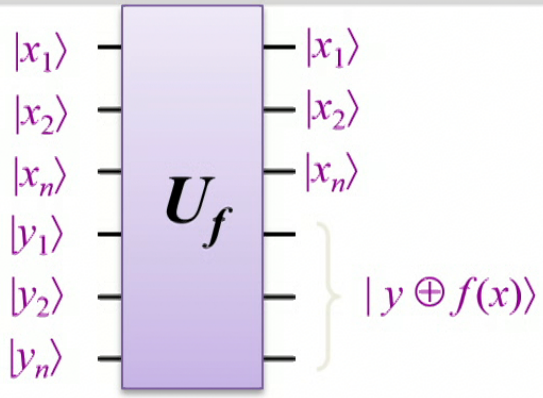
Queries:



22

A quantum algorithm for Simon (1)

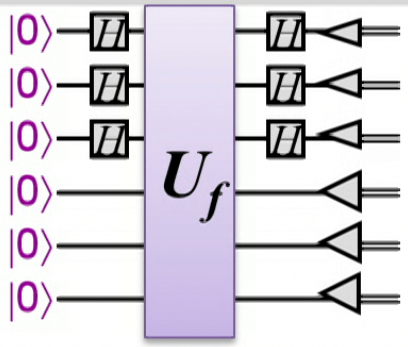
Queries:



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

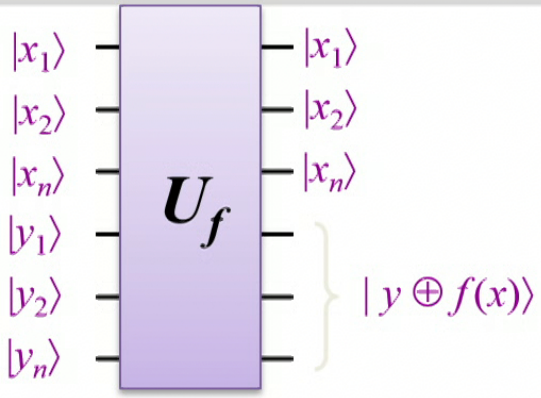
A quantum algorithm for Simon (4)

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



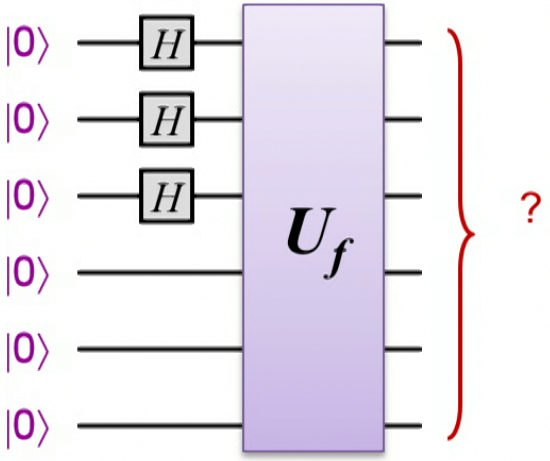
A quantum algorithm for Simon (1)

Queries:



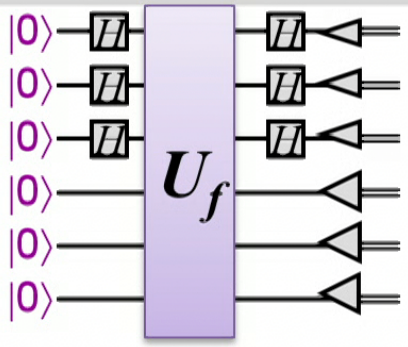
Proposed start of quantum algorithm: query all values of f in superposition

What is the output state of this circuit?



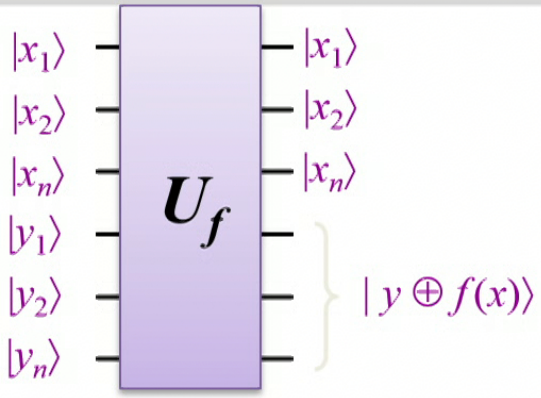
A quantum algorithm for Simon (4)

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



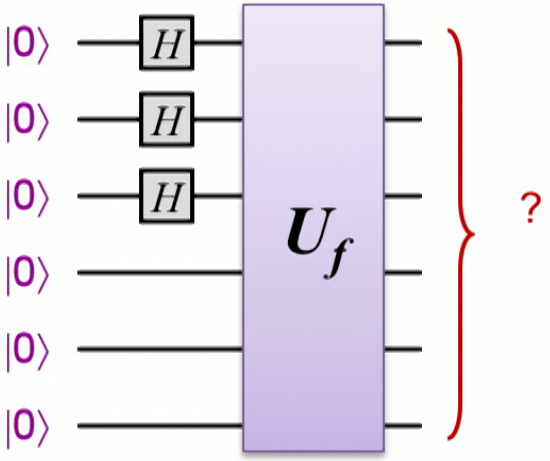
A quantum algorithm for Simon (1)

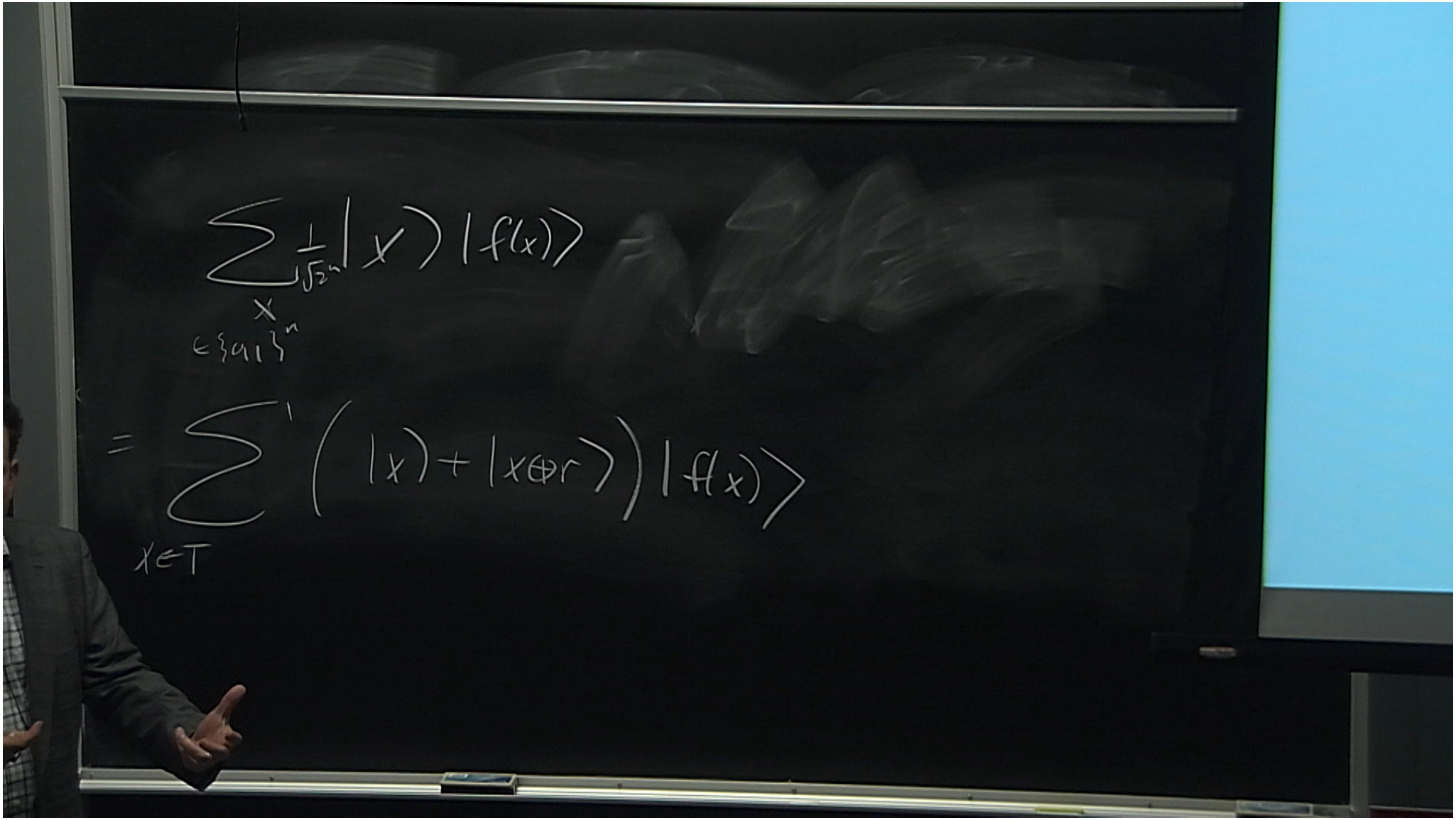
Queries:

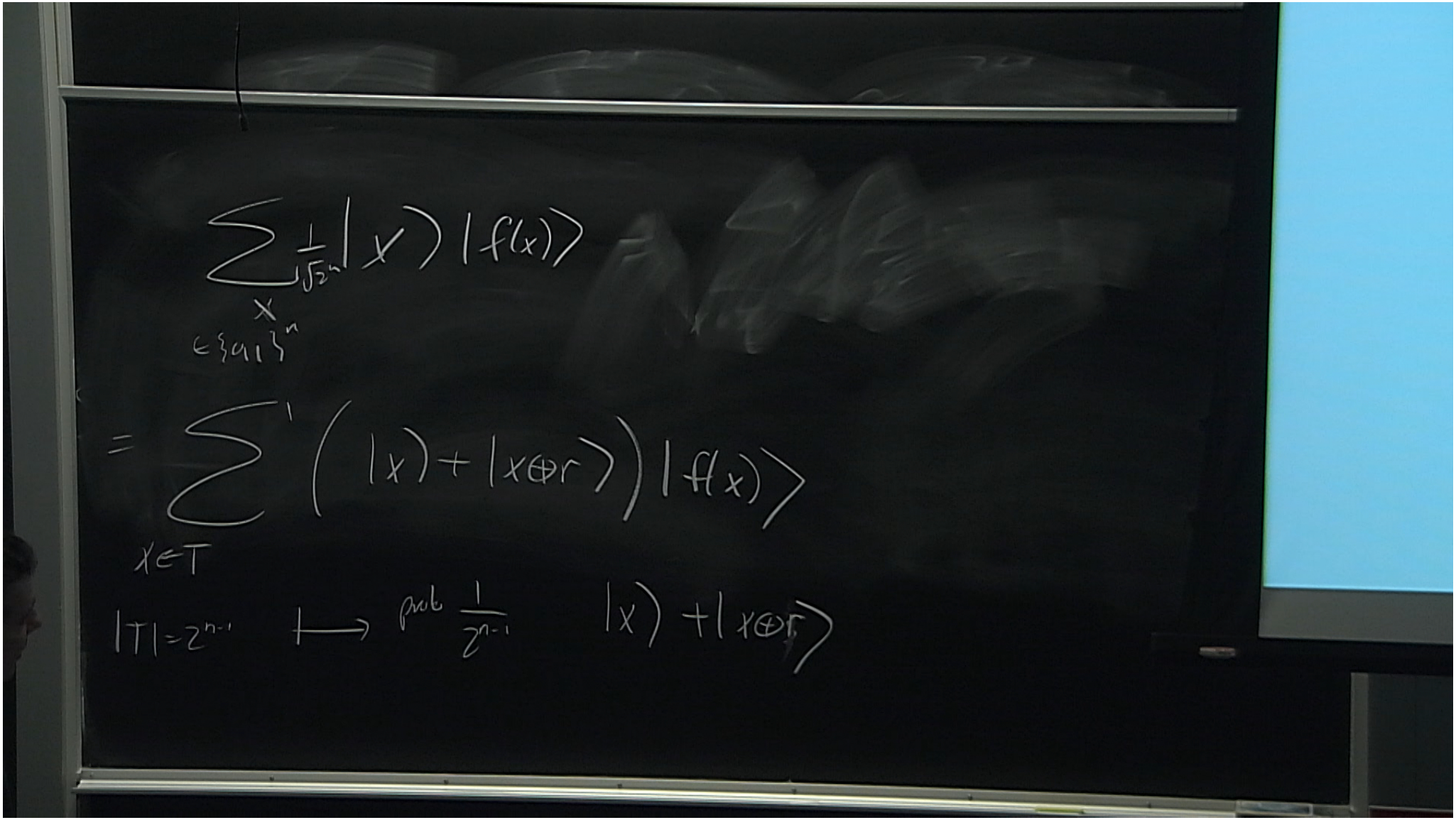


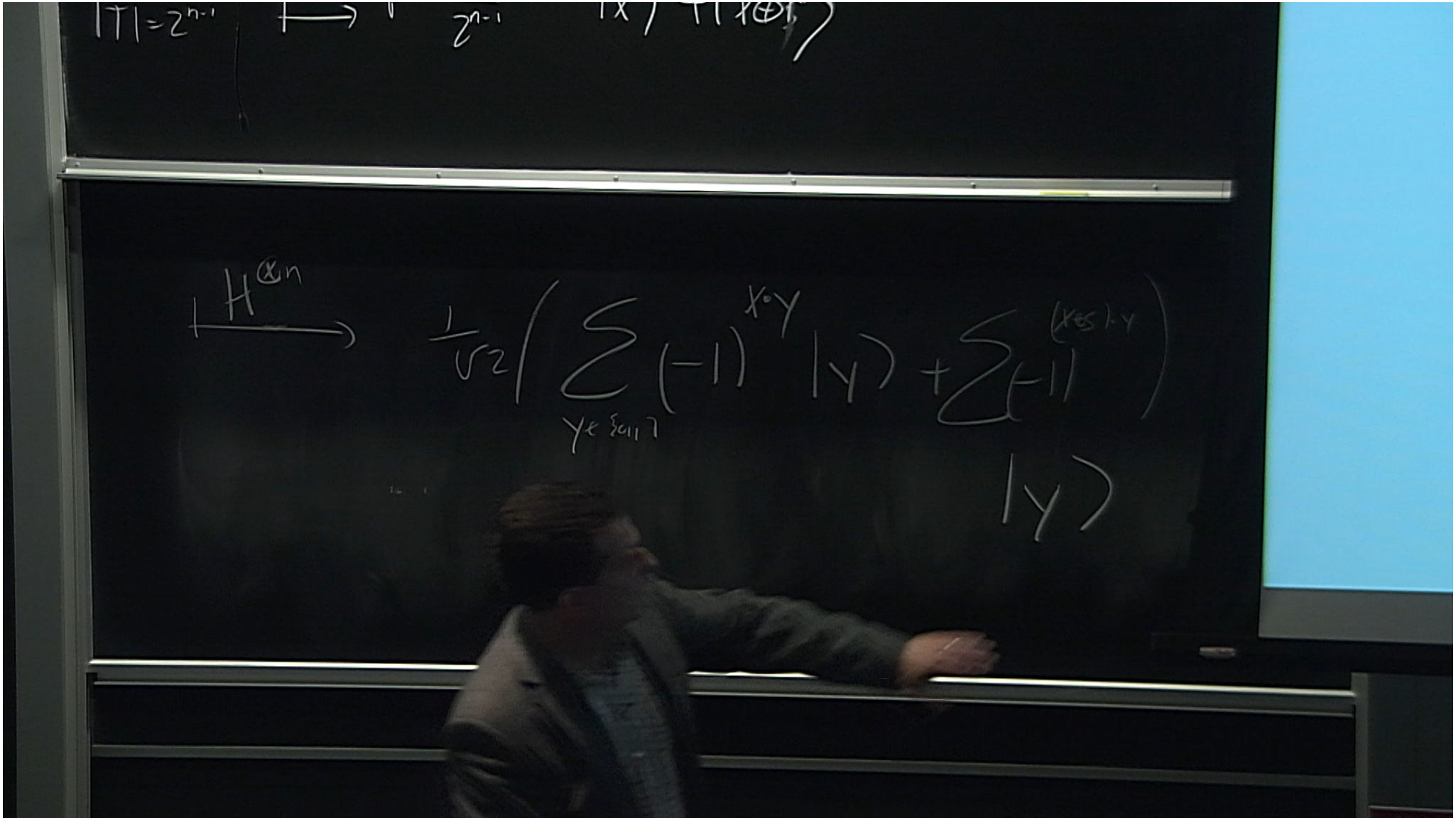
Proposed start of quantum algorithm: query all values of f in superposition

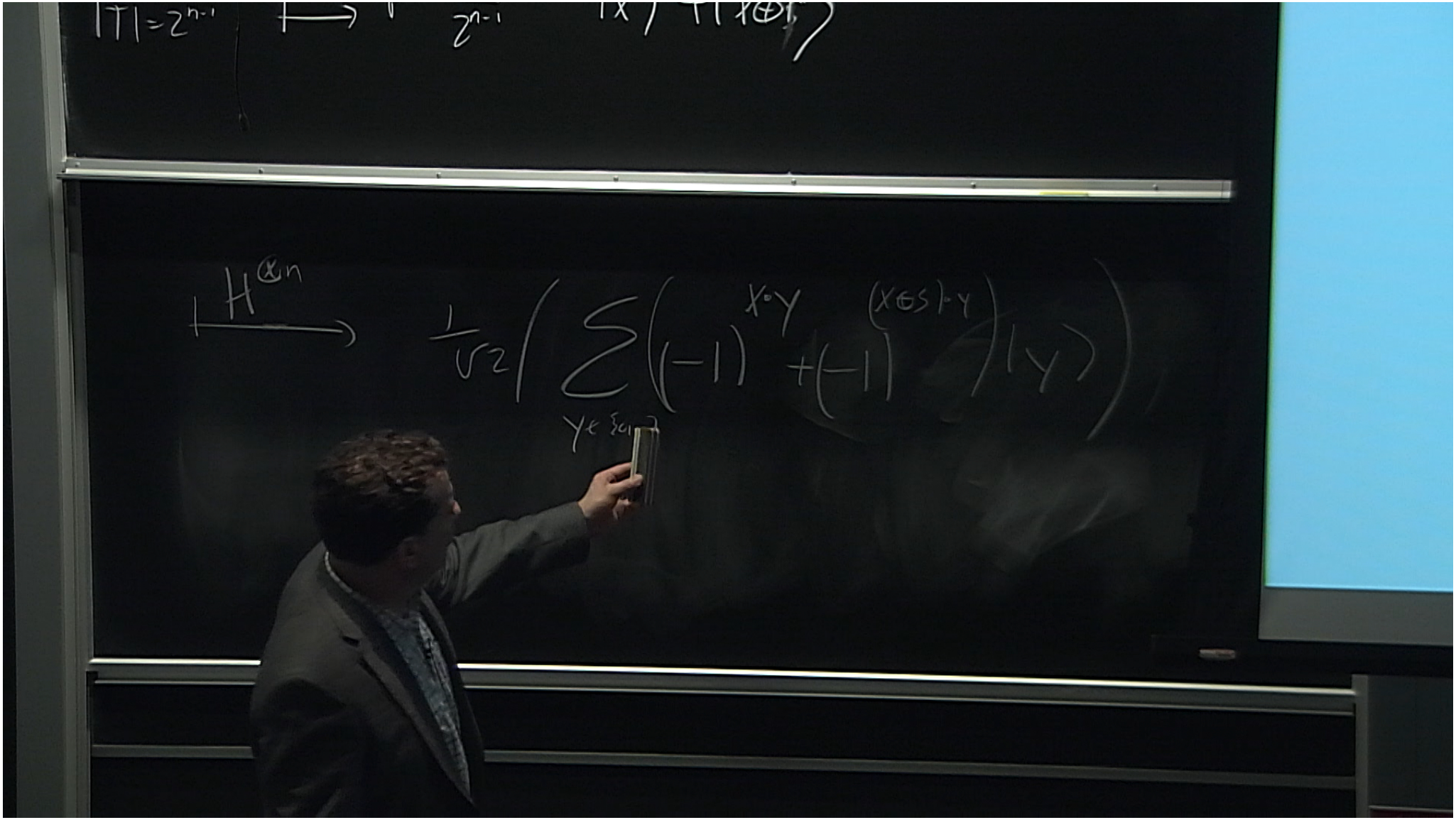
What is the output state of this circuit?

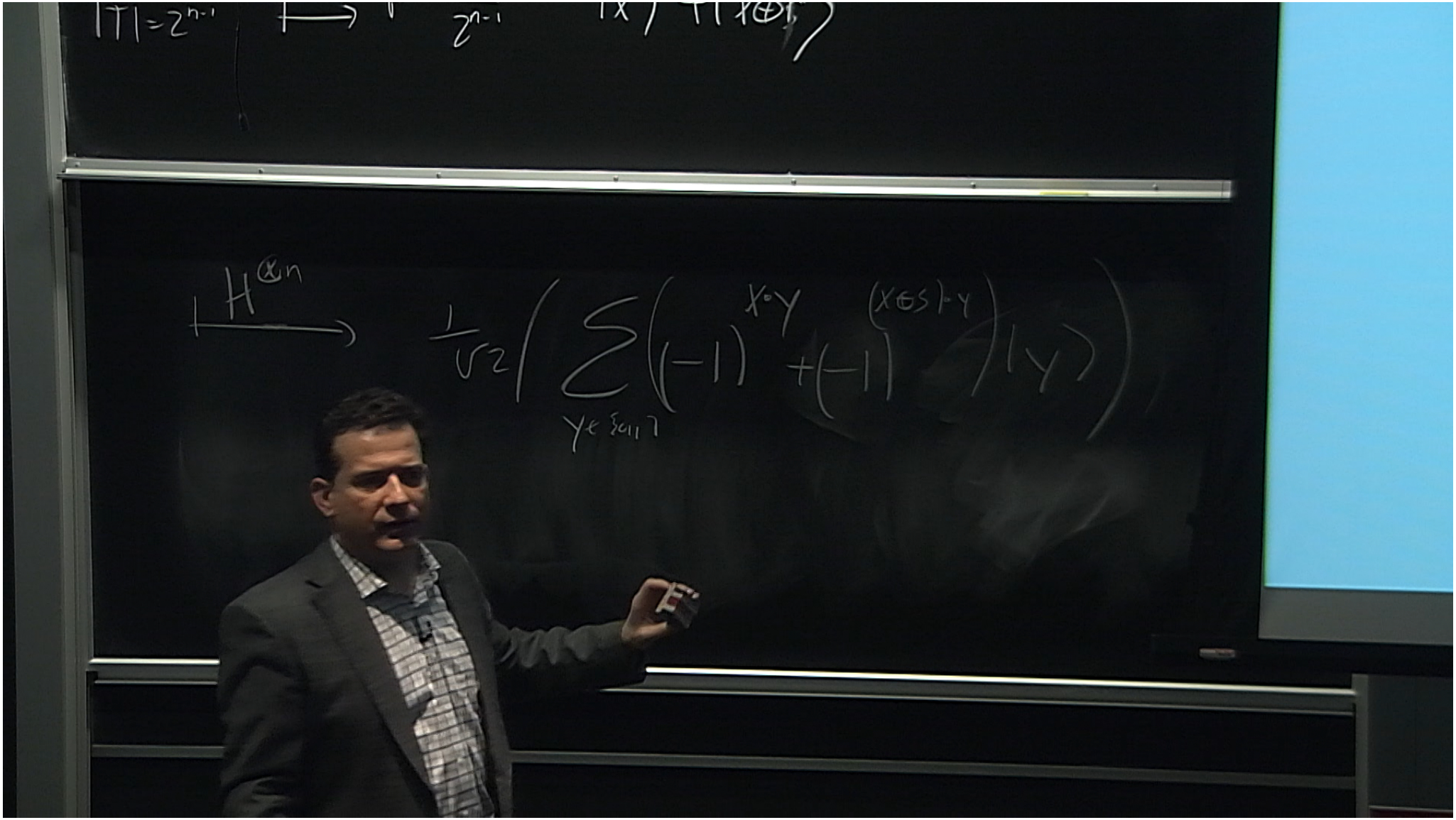


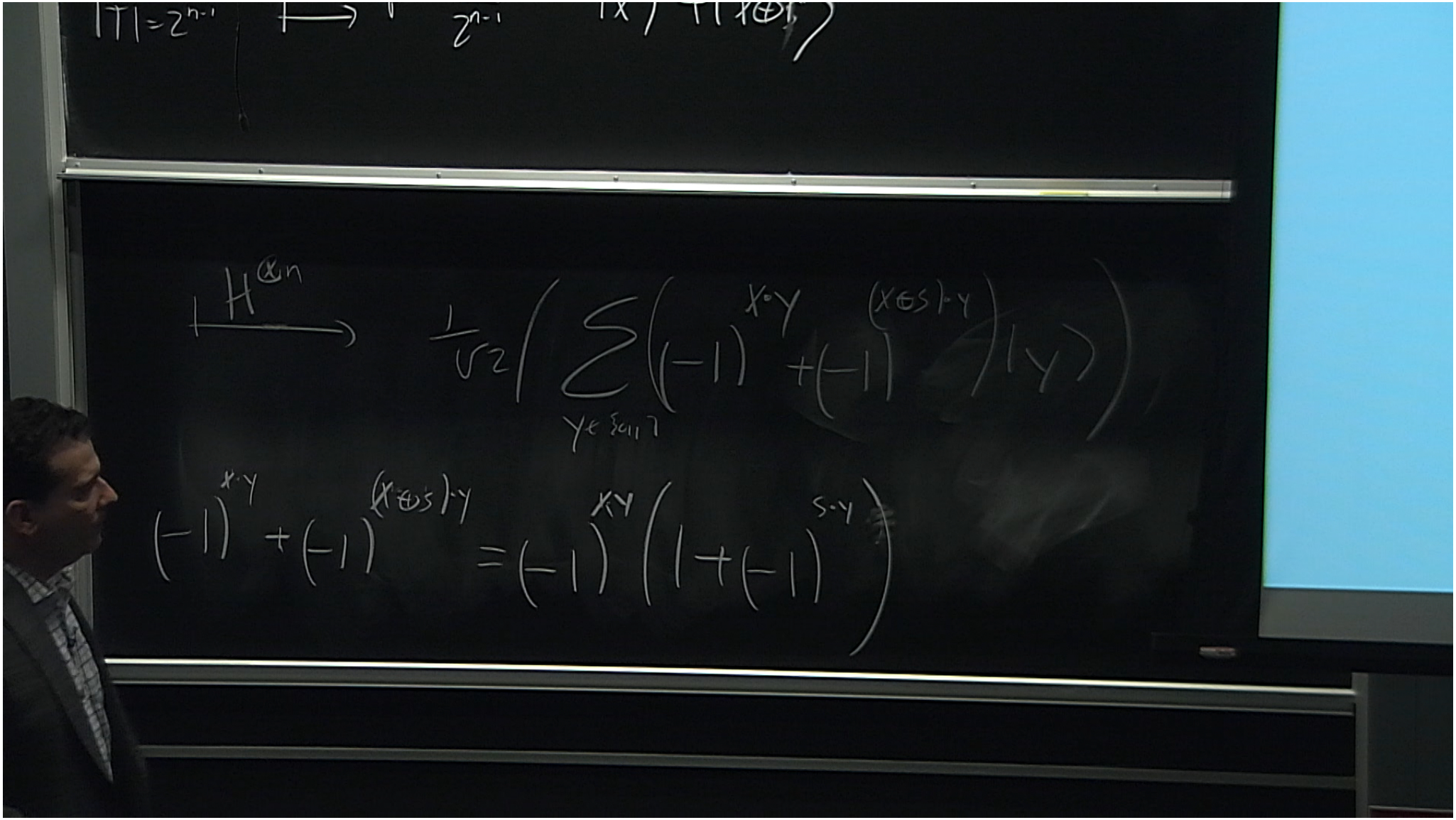

$\sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}} |x\rangle |f(x)\rangle$
$= \sum_{x \in T} \frac{1}{\sqrt{2}} \left(|x\rangle + |x \oplus r\rangle \right) |f(x)\rangle$

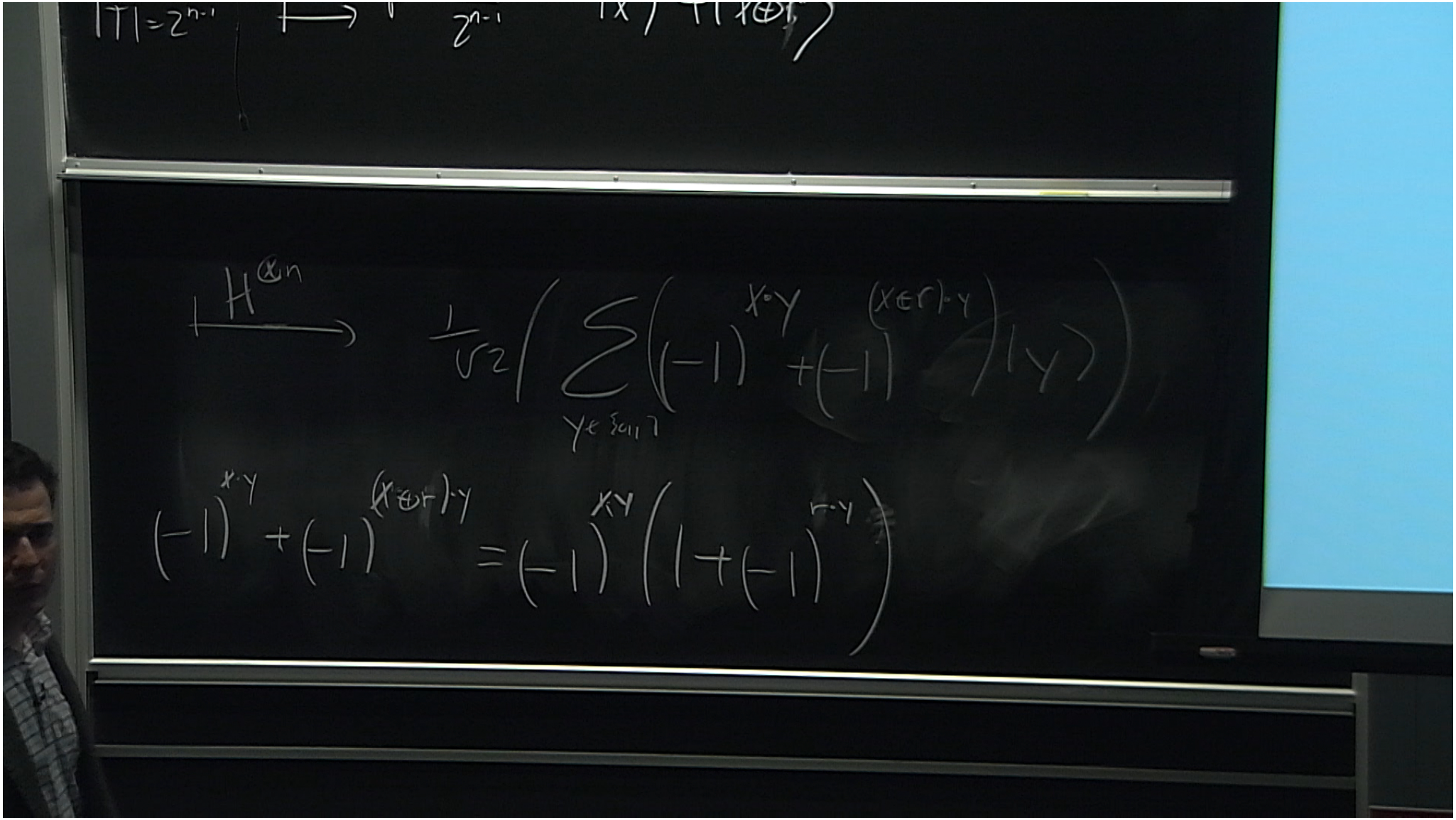

$\sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}} |x\rangle |f(x)\rangle$
$= \sum_{x \in T}' \left(|x\rangle + |x \oplus r\rangle \right) |f(x)\rangle$
$|T| = 2^{n-1}$
$\longmapsto$ prob $\frac{1}{2^{n-1}}$
$|x\rangle + |x \oplus r\rangle$

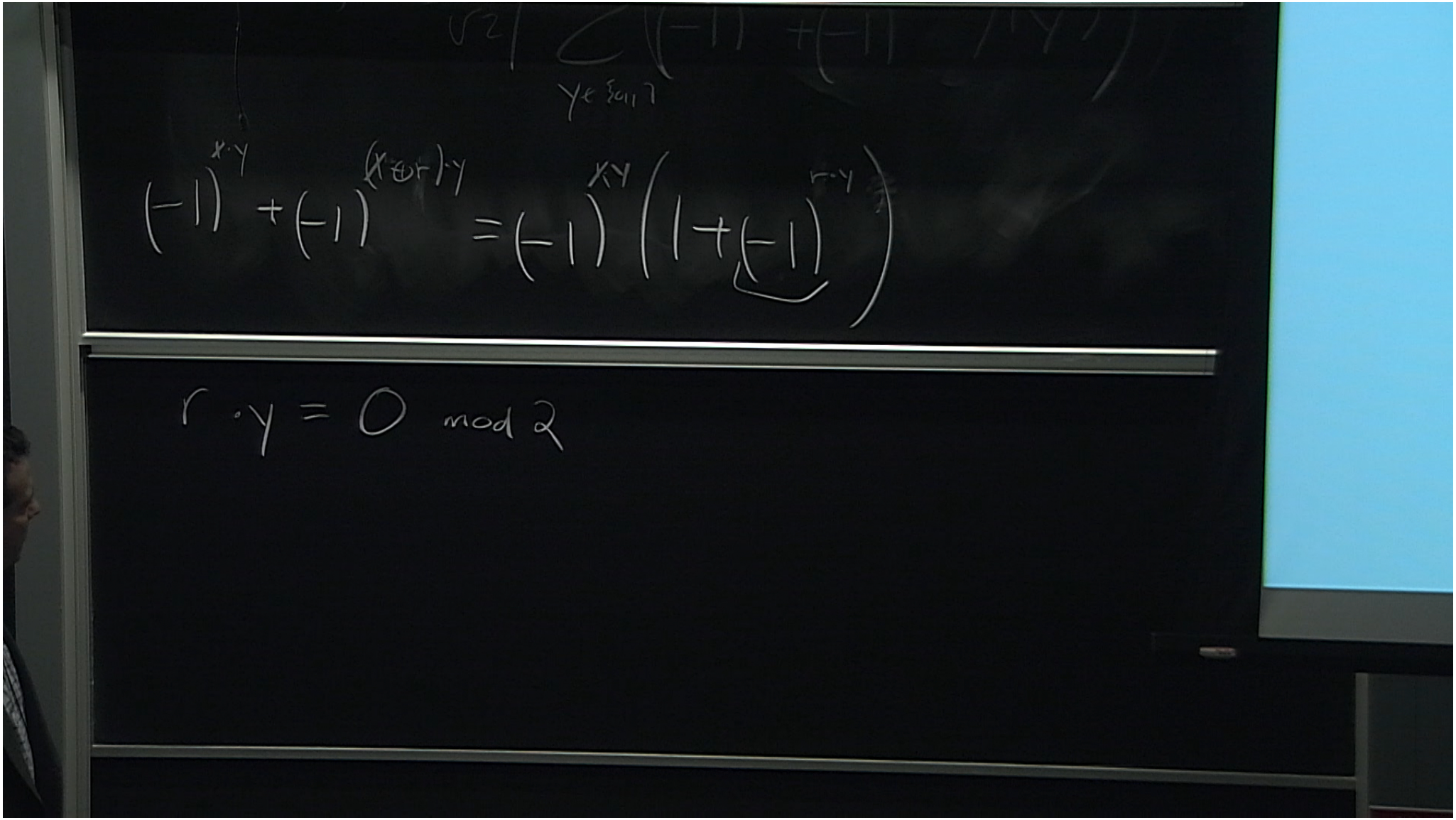

$|T| = 2^{n-1}$
2^{n-1}
$H^{\otimes n}$
$\frac{1}{\sqrt{2}}\left(\sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle + \sum (-1)^{(x \in S) \cdot y} |y\rangle\right)$


$|T| = 2^{n-1}$
2^{n-1}
$H^{\otimes n}$
$\frac{1}{\sqrt{2}}\left(\sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} + (-1)^{(x \oplus s) \cdot y} |y\rangle\right)$


$|T| = 2^{n-1}$
$\longmapsto$
2^{n-1}
$H^{\otimes n}$
$\frac{1}{\sqrt{2}}\left(\sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} + (-1)^{(x \oplus s) \cdot y}\right)|y\rangle$

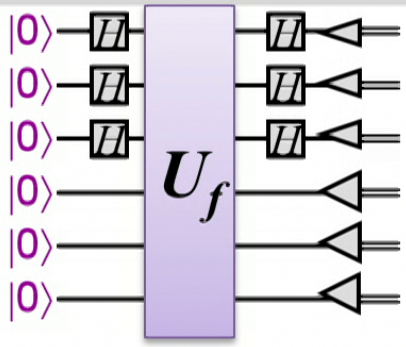

$|T| = 2^{n-1}$
$\longmapsto$
2^{n-1}
$H^{\otimes n}$
$\frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}^n} \left((-1)^{x \cdot y} + (-1)^{(x \oplus s) \cdot y} \right) |y\rangle$
$(-1)^{x \cdot y} + (-1)^{(x \oplus s) \cdot y} = (-1)^{x \cdot y} \left(1 + (-1)^{s \cdot y} \right)$


$|T| = 2^{n-1}$
$\longmapsto$
2^{n-1}
$H^{\otimes n}$
$\longmapsto$
$\frac{1}{\sqrt{2}}\left(\sum_{y \in \{0,1\}^n} \left((-1)^{x \cdot y} + (-1)^{(x \oplus r) \cdot y}\right)|y\rangle\right)$
$(-1)^{x \cdot y} + (-1)^{(x \oplus r) \cdot y} = (-1)^{x \cdot y}\left(1 + (-1)^{r \cdot y}\right)$


$y \in \{0,1\}$
$(-1)^{x \cdot y} + (-1)^{(x \oplus r) \cdot y} = (-1)^{xy}\left(1 + (-1)^{r \cdot y}\right)$
$r \cdot y = 0 \mod 2$

A quantum algorithm for Simon (4)

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:

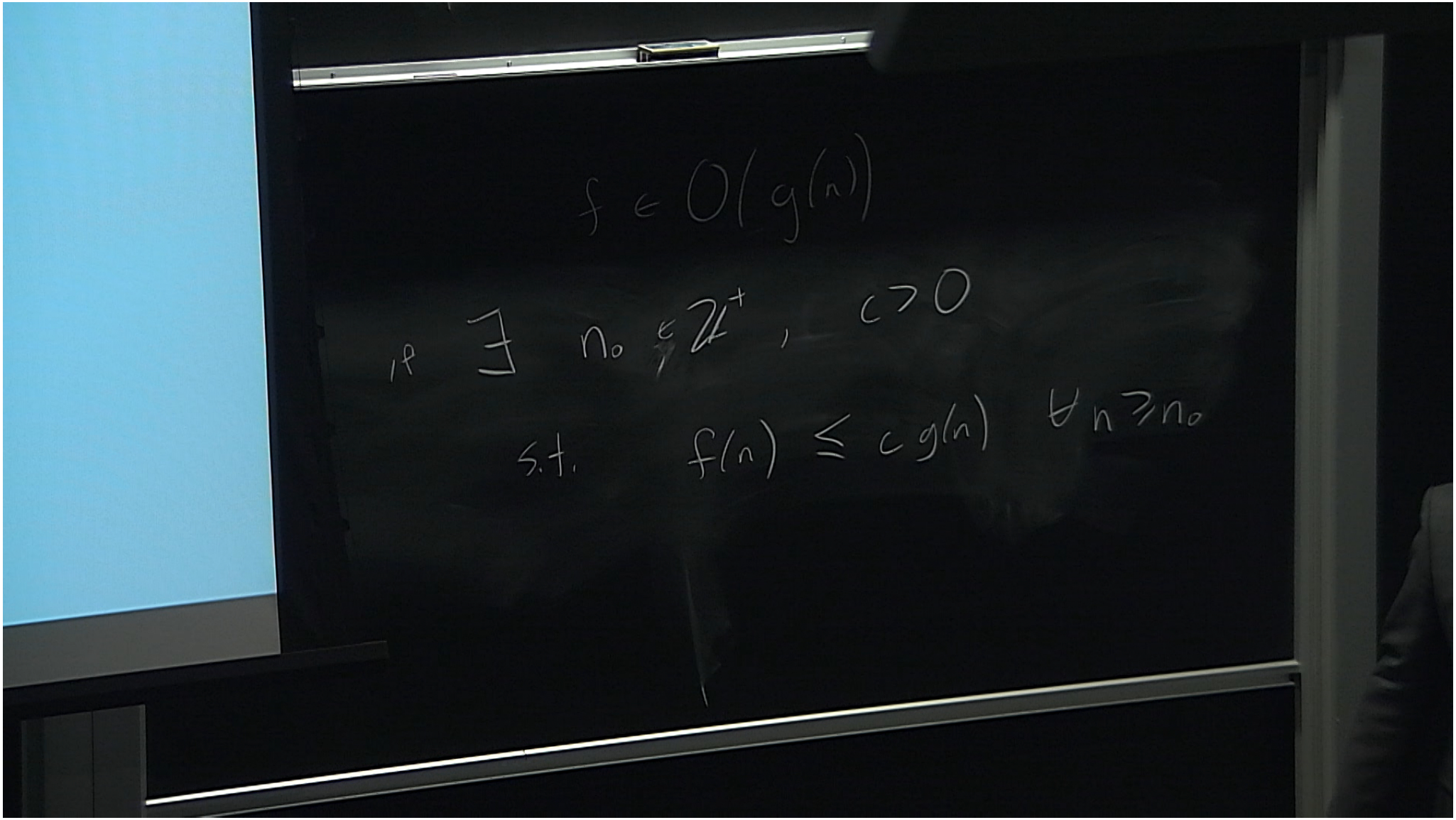
$$\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{k1} & y_{k2} & \cdots & y_{kn} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

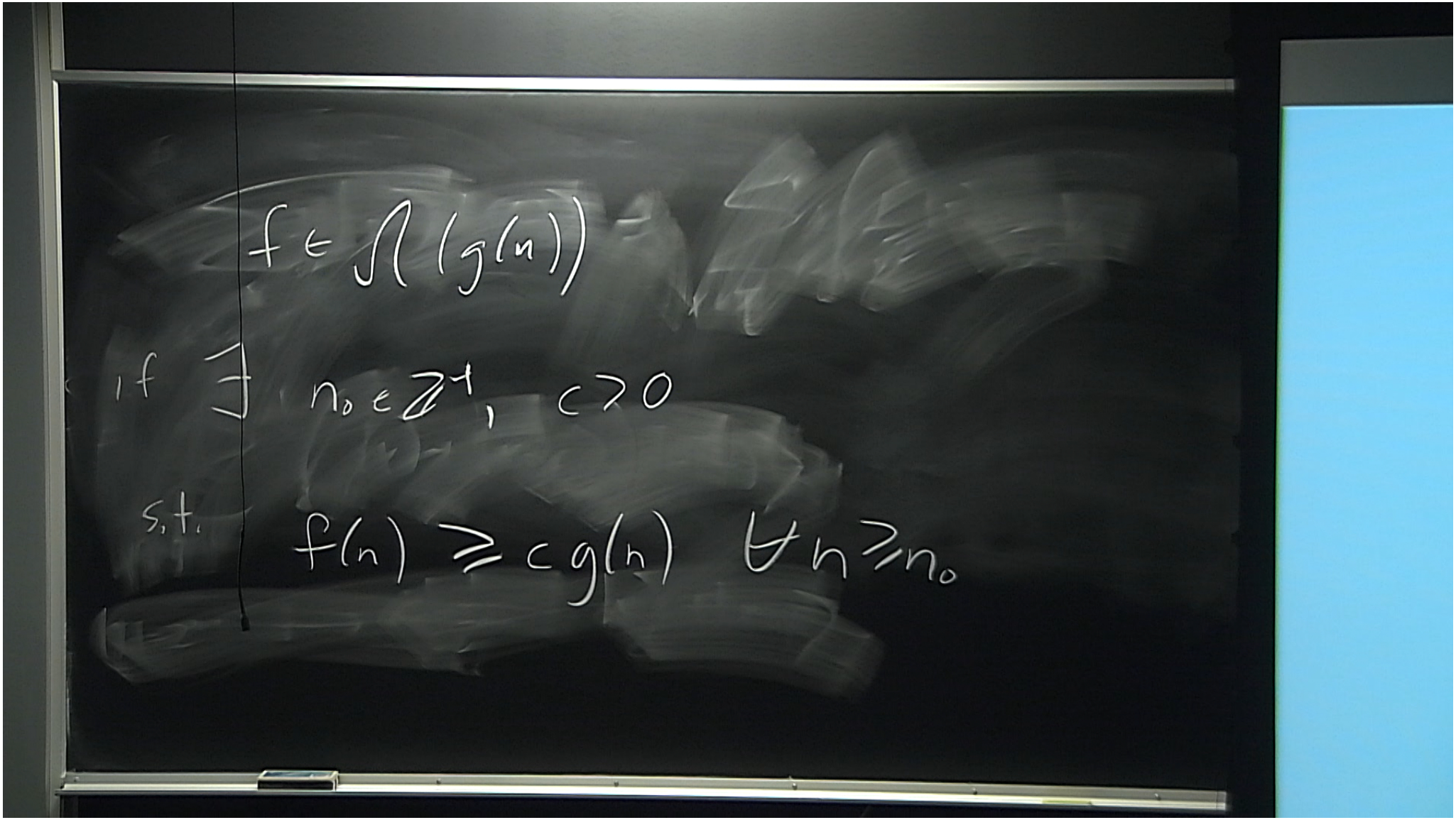
With high probability, there is a unique non-zero solution that is r (which can be efficiently found by linear algebra)

Conclusion of Simon's algorithm

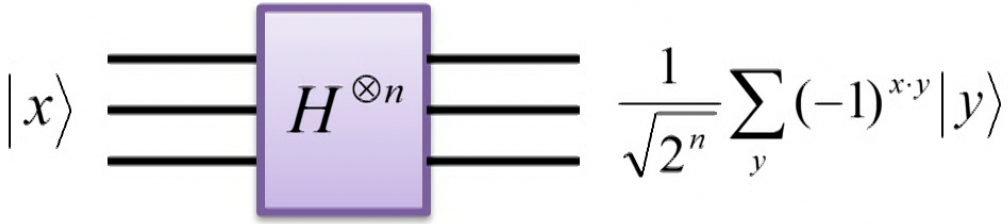
- Any classical algorithm has to query the black box $\Omega(2^{n/2})$ times, even to succeed with probability ¾.

- 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.

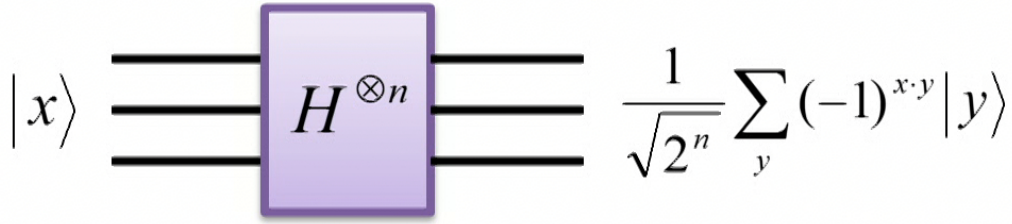

$f \in O(g(n))$
if $\exists\ n_0 \in \mathbb{Z}^+$, $c > 0$
s.t. $f(n) \leq c\,g(n)$ $\forall n \geq n_0$

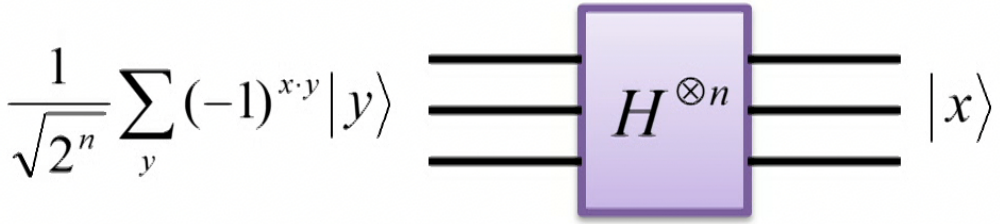

$f \in \Omega(g(n))$
if $\exists\ n_0 \in \mathbb{Z}^+,\ c > 0$
s.t. $f(n) \geq c\,g(n) \quad \forall n \geq n_0$

Multi-qubit Hadamard



Multi-qubit Hadamard





Quantum algorithms

- The previously mentioned algorithms have been computing essentially classical functions on quantum superpositions
- This encoded information in the phases of the basis states: measuring basis states would provide little useful information

Quantum phase estimation

- Suppose we wish to estimate a number $\omega \in [0,1)$ given the quantum state

$$\sum_{y=0}^{2^n-1} e^{2\pi i \omega y} |y\rangle$$

Quantum phase estimation

- Suppose we wish to estimate a number $\omega \in [0,1)$ given the quantum state

$$\sum_{y=0}^{2^n-1} e^{2\pi i \omega y} |y\rangle$$

- Note that in binary we can express

$$\omega = 0.x_1 x_2 x_3 \ldots$$

$$2\omega = x_1 . x_2 x_3 \ldots$$

$$2^{n-1}\omega = x_1 x_2 x_3 \ldots x_{n-1} . x_n x_{n+1} \ldots$$

Quantum phase estimation

- Since $e^{2\pi ik} = 1$ for any integer *k*, we have

$$e^{2\pi i(2\omega)} = e^{2\pi i(x_1.x_2x_3\ldots)} = e^{2\pi i x_1} e^{2\pi i(0.x_2x_3\ldots)} = e^{2\pi i(0.x_2x_3\ldots)}$$

$$e^{2\pi i(2^k\omega)} = e^{2\pi i(0.x_{k+1}x_{k+2}\ldots)}$$

Quantum phase estimation

- If $\omega = 0.x_1$ then we can do the following

Quantum phase estimation

- If $\omega = 0.x_1$ then we can do the following

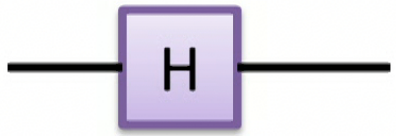
$$\frac{|0\rangle + e^{2\pi i(0.x_1)}|1\rangle}{\sqrt{2}} = \frac{|0\rangle + (-1)^{x_1}|1\rangle}{\sqrt{2}}$$

Quantum phase estimation

- If $\omega = 0.x_1$ then we can do the following

$$\frac{|0\rangle + e^{2\pi i(0.x_1)}|1\rangle}{\sqrt{2}} = \frac{|0\rangle + (-1)^{x_1}|1\rangle}{\sqrt{2}}$$

$$\frac{|0\rangle + (-1)^{x_1}|1\rangle}{\sqrt{2}}$$

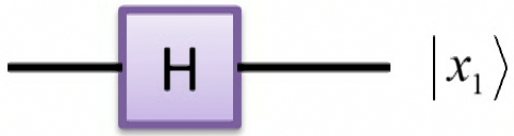


Quantum phase estimation

- If $\omega = 0.x_1$ then we can do the following

$$\frac{|0\rangle + e^{2\pi i(0.x_1)}|1\rangle}{\sqrt{2}} = \frac{|0\rangle + (-1)^{x_1}|1\rangle}{\sqrt{2}}$$

$$\frac{|0\rangle + (-1)^{x_1}|1\rangle}{\sqrt{2}}$$



Useful identity

- We can show that

$$\sum_{y=0}^{2^n-1} e^{2\pi i \omega y} |y\rangle$$

Useful identity

- We can show that

$$\sum_{y=0}^{2^n-1} e^{2\pi i \omega y} |y\rangle$$

$$= \left(|0\rangle + e^{2\pi i (2^{n-1}\omega)}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i (2^{n-2}\omega)}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i (\omega)}|1\rangle\right)$$

Useful identity

- We can show that

$$\sum_{y=0}^{2^n-1} e^{2\pi i \omega y} |y\rangle$$

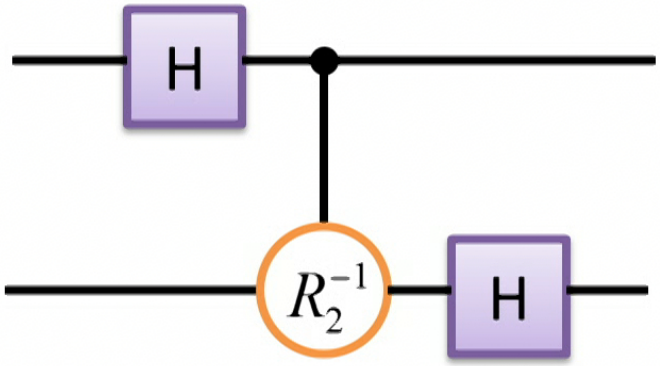
$$= \left(|0\rangle + e^{2\pi i (2^{n-1}\omega)}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i (2^{n-2}\omega)}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i (\omega)}|1\rangle\right)$$

$$= \left(|0\rangle + e^{2\pi i (0.x_n x_{n+1} \ldots)}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i (0.x_{n-1} x_n x_{n+1} \ldots)}|1\rangle\right) \otimes$$
$$\cdots \otimes \left(|0\rangle + e^{2\pi i (0.x_1 x_2 \ldots)}|1\rangle\right)$$

Quantum phase estimation

- So if $\omega = 0.x_1x_2$ then we can do the following

$$\frac{|0\rangle + e^{2\pi i(0.x_2)}|1\rangle}{\sqrt{2}}$$

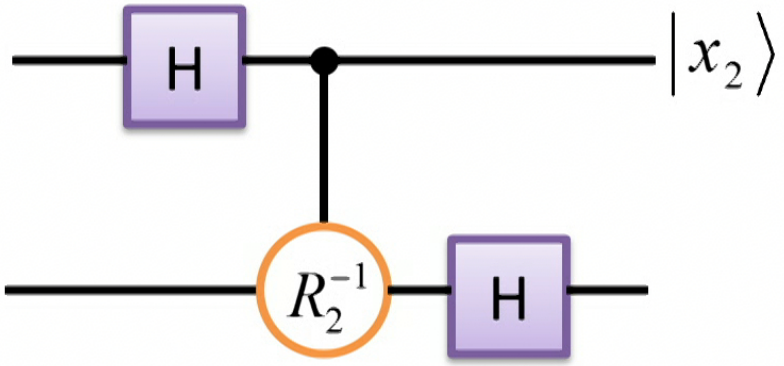


Quantum phase estimation

- So if $\omega = 0.x_1x_2$ then we can do the following

$$\frac{|0\rangle + e^{2\pi i(0.x_2)}|1\rangle}{\sqrt{2}}$$

$$\frac{|0\rangle + e^{2\pi i(0.x_1x_2)}|1\rangle}{\sqrt{2}}$$

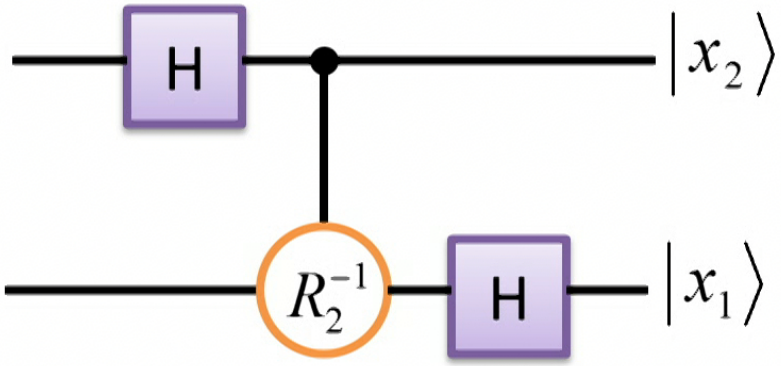


Quantum phase estimation

- So if $\omega = 0.x_1x_2$ then we can do the following

$$\frac{|0\rangle + e^{2\pi i(0.x_2)}|1\rangle}{\sqrt{2}}$$

$$\frac{|0\rangle + e^{2\pi i(0.x_1x_2)}|1\rangle}{\sqrt{2}}$$

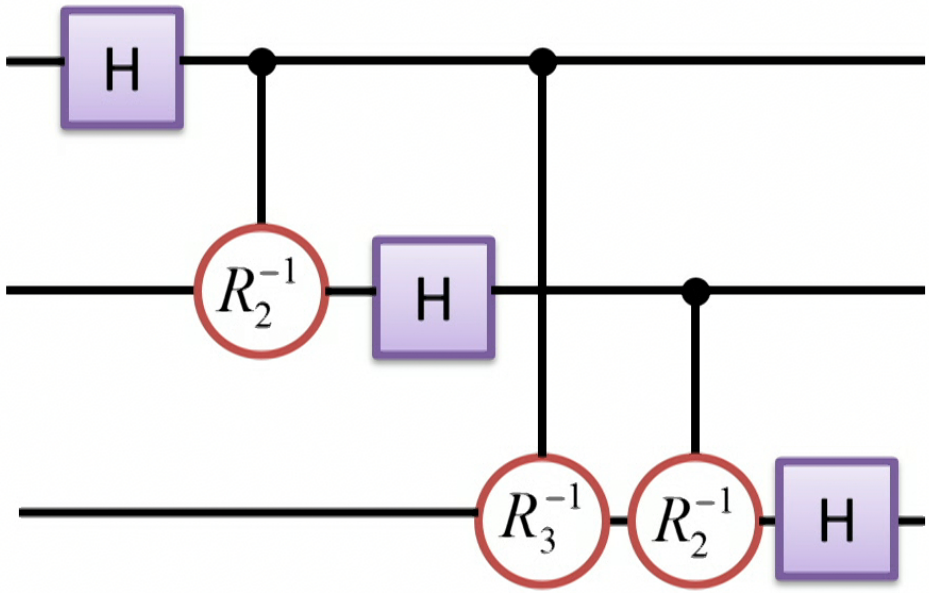


$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

Quantum phase estimation

- So if $\omega = 0.x_1x_2x_3$ then we can do the following

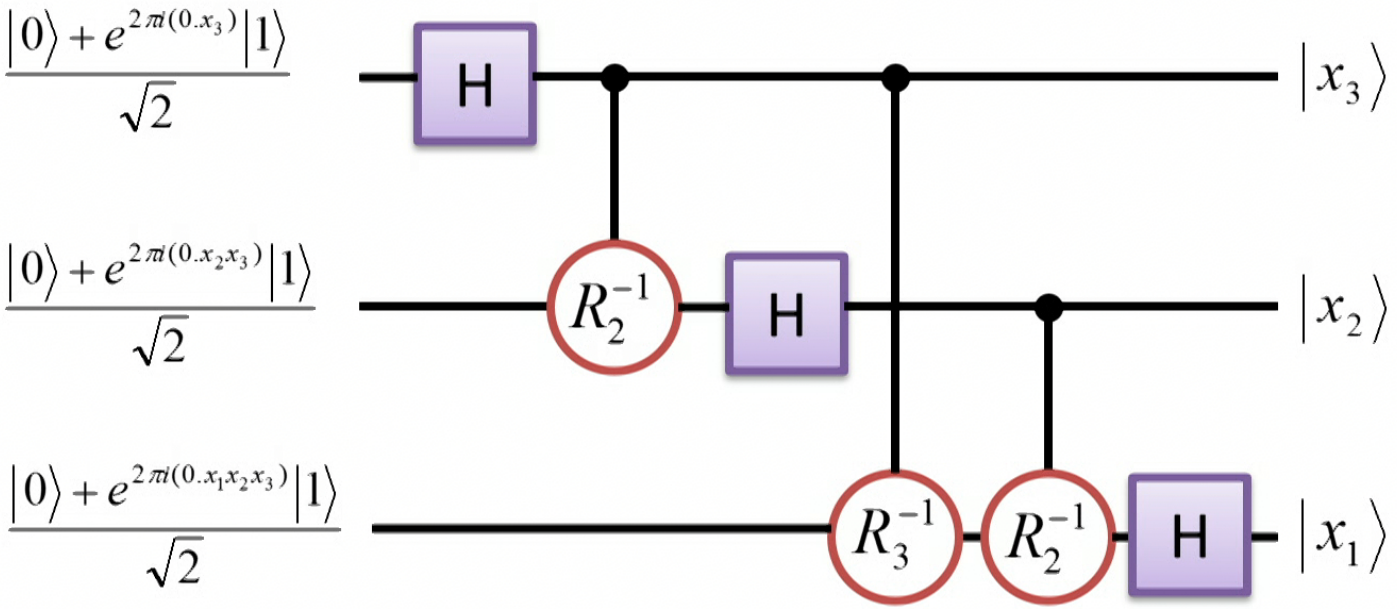
$$\frac{|0\rangle + e^{2\pi i(0.x_3)}|1\rangle}{\sqrt{2}}$$



32

Quantum phase estimation

- So if $\omega = 0.x_1x_2x_3$ then we can do the following



Quantum phase estimation

- Generalizing this network (and reversing the order of the qubits at the end) gives us a network with $O(n^2)$ gates that implements

$$\sum_{y=0}^{2^n-1} e^{2\pi i \frac{x}{2^n} y} |y\rangle \rightarrow |x\rangle$$

Discrete Fourier transform

- The discrete Fourier transform maps vectors of dimension *N* by transforming the x^{th} elementary vector according to

$$(0,0,\ldots,0,1,0,\ldots 0) \rightarrow (1, e^{2\pi i \frac{x}{N}}, e^{2\pi i \frac{2x}{N}}, \ldots, e^{2\pi i \frac{(N-1)x}{N}})$$

Discrete Fourier transform

- The discrete Fourier transform maps vectors of dimension *N* by transforming the x^{th} elementary vector according to

$$(0,0,...,0,1,0,...0) \rightarrow (1, e^{2\pi i \frac{x}{N}}, e^{2\pi i \frac{2x}{N}}, \ldots, e^{2\pi i \frac{(N-1)x}{N}})$$

- The quantum Fourier transform maps vectors in a Hilbert space of dimension *N* according to

$$|x\rangle \rightarrow \sum_{y=0}^{N-1} e^{2\pi i \frac{x}{N} y} |y\rangle$$

Discrete Fourier transform

- Thus we have illustrated how to implement (the inverse of) the quantum Fourier transform in a Hilbert space of dimension 2^n

Estimating arbitrary ω ∈ [0,1)

- What if ω is not necessarily of the form $\frac{x}{2^n}$ for some integer x ?

- The QFT will map

$$\sum_{x=0}^{2^n-1} e^{2\pi i \omega z} \left| z \right\rangle$$

Estimating arbitrary ω ∈ [0,1)

- What if ω is not necessarily of the form $\frac{x}{2^n}$ for some integer x ?

- The QFT will map

$$\sum_{x=0}^{2^n-1} e^{2\pi i \omega z} |z\rangle$$

to a superposition

$$|\tilde{\omega}\rangle = \sum_y \alpha_y |y\rangle$$

Estimating arbitrary ω ∈ [0,1)

- What if ω is not necessarily of the form $\frac{x}{2^n}$ for some integer *x* ?

- The QFT will map

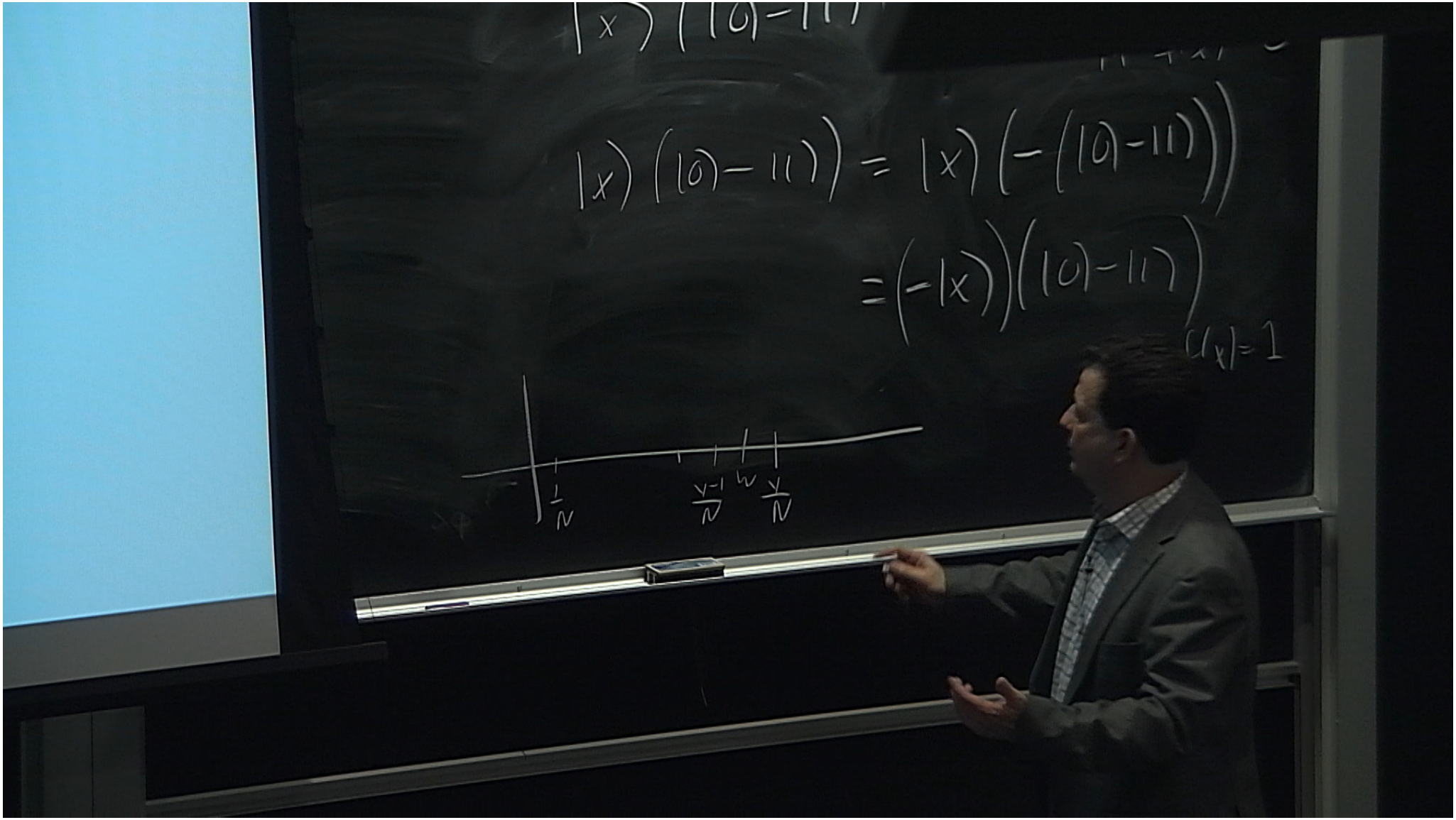
$$\sum_{x=0}^{2^n-1} e^{2\pi i \omega z} |z\rangle$$

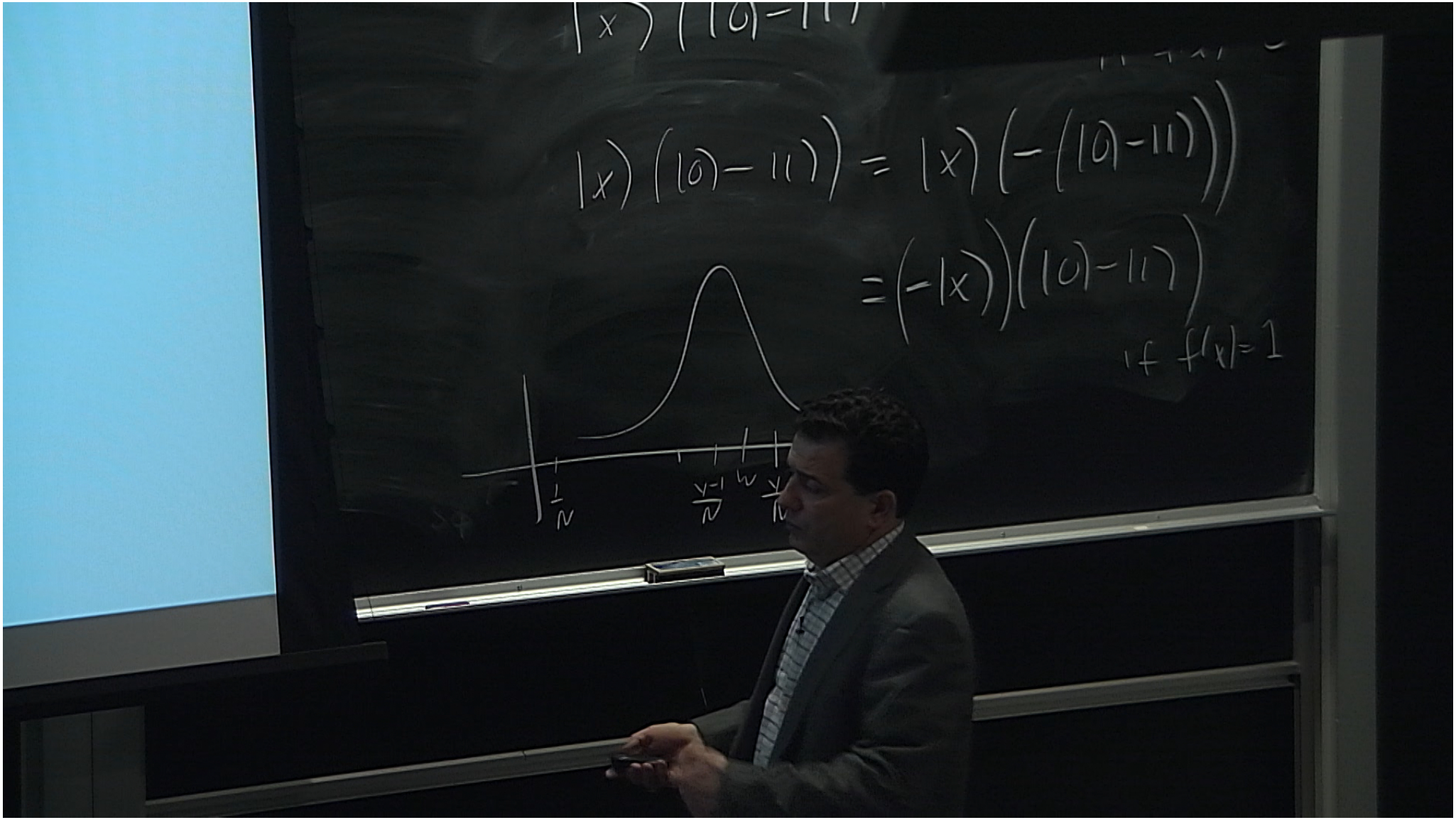
to a superposition

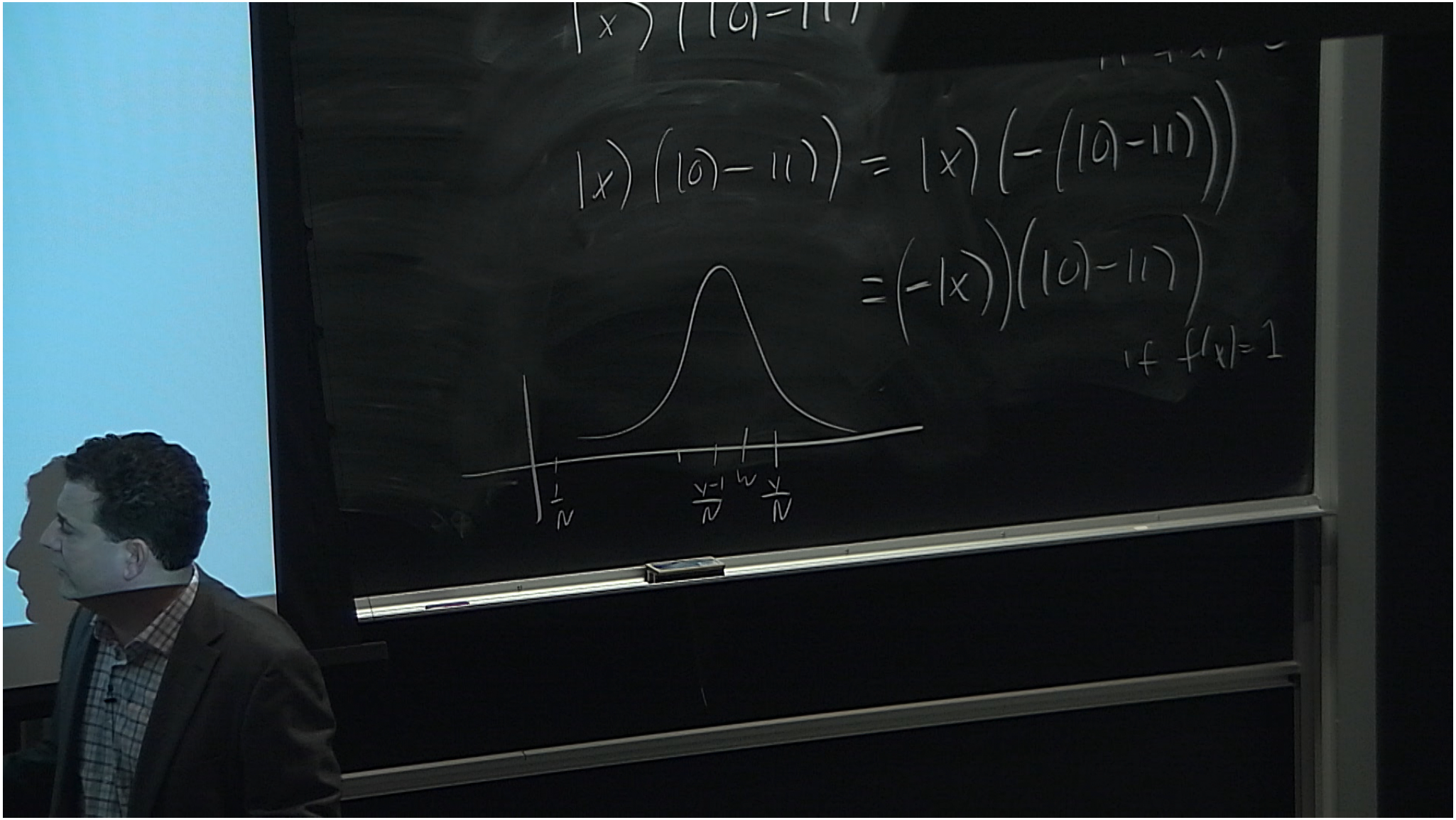
$$|\tilde{\omega}\rangle = \sum_y \alpha_y |y\rangle$$

where

$$\Pr ob\left(\left|\frac{y}{N} - \omega\right| \le \frac{1}{N}\right) \ge \frac{8}{\pi^2}$$


|x)(|0⟩−|1⟩)
|x)(|0⟩−|1⟩) = |x)(−(|0⟩−|1⟩))
= (−|x))(|0⟩−|1⟩)
C(x) = 1
1/N
N−1/N
N/N


|x)(|0)-|1)|
|x)(|0)-|1)| = |x)(-(|0)-|1)|)
=(-|x))(|0)-|1)|)
if f(x)=1


|x)(|0)-|1|)
|x)(|0)-|1|) = |x)(-(|0)-|1|))
=(-|x))(|0)-|1|)
if f(x)=1
1/N
N-1/N
N/N

Estimating arbitrary ω ∈ [0,1)

- What if ω is not necessarily of the form $\frac{x}{2^n}$ for some integer x ?

- The QFT will map

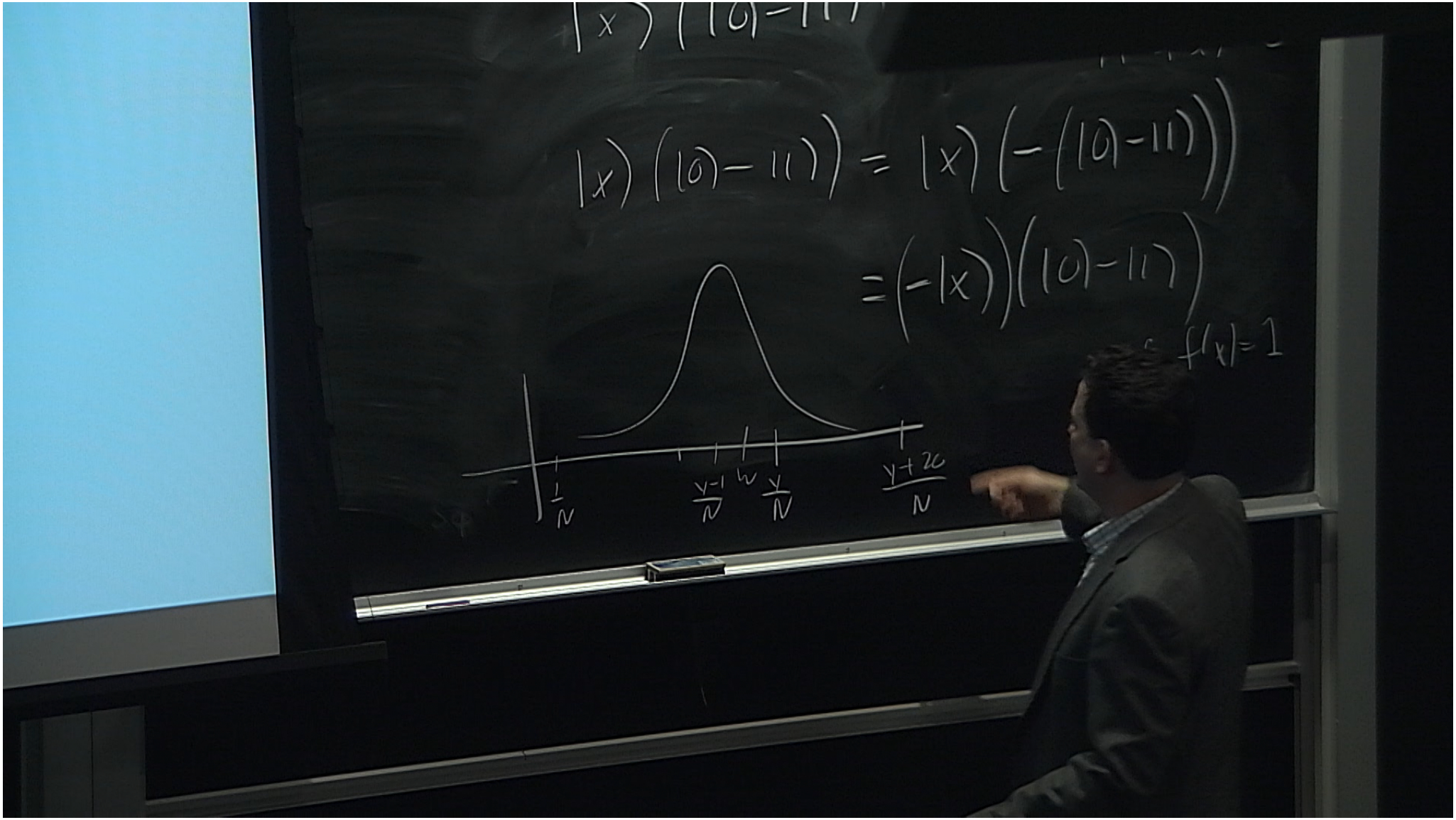
$$\sum_{x=0}^{2^n-1} e^{2\pi i \omega z} |z\rangle$$

to a superposition

$$|\tilde{\omega}\rangle = \sum_y \alpha_y |y\rangle$$

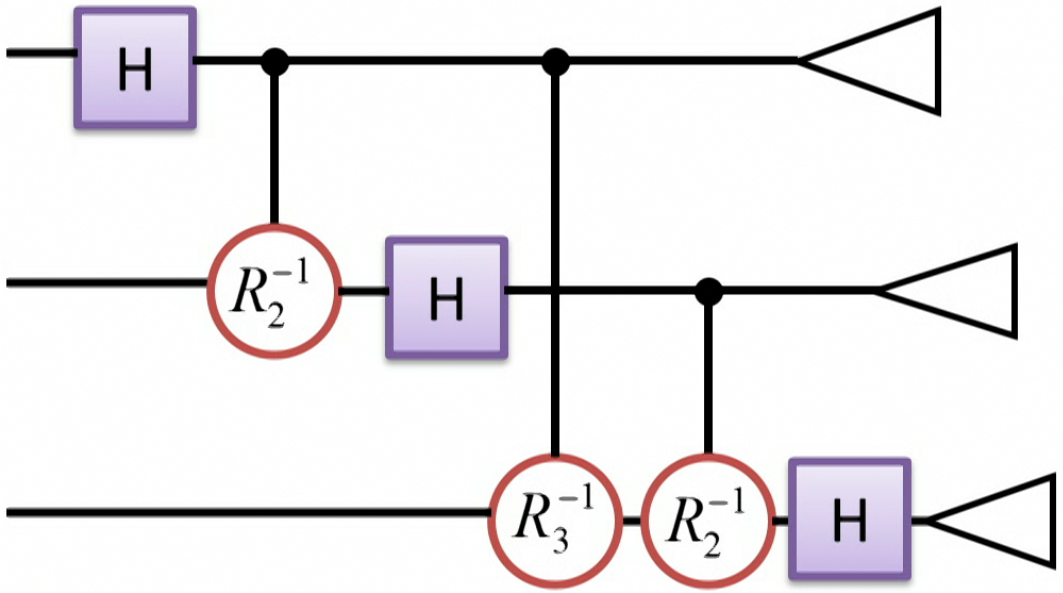
where

$$\Pr ob\left(\left|\frac{y}{N} - \omega\right| \le \frac{1}{N}\right) \ge \frac{8}{\pi^2} \qquad |\alpha_y| \in O\left(\frac{1}{\left|\frac{y}{N} - \omega\right|}\right)$$


|x)(10-11)
|x)(10-11) = |x)(-(10-11))
=(-|x))(10-11)
f(x)=1
1/N
y-1/N
y/N
y+2c/N

Quantum phase estimation

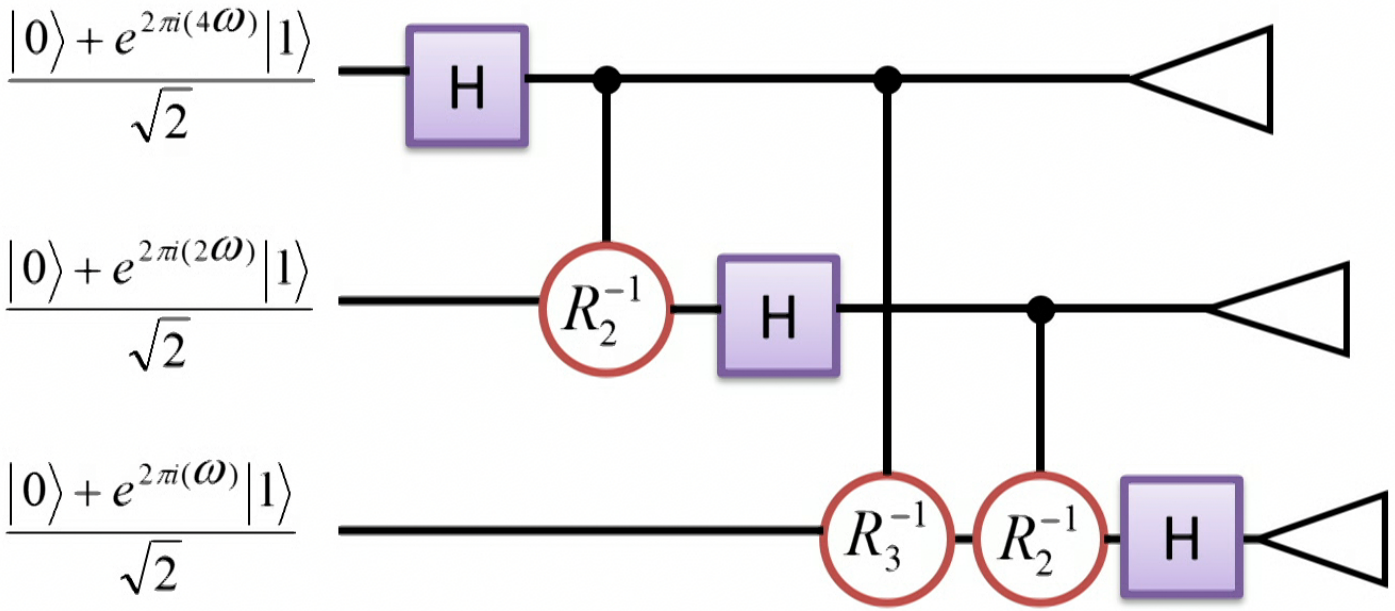
- For any real $\omega \in [0,1)$



37

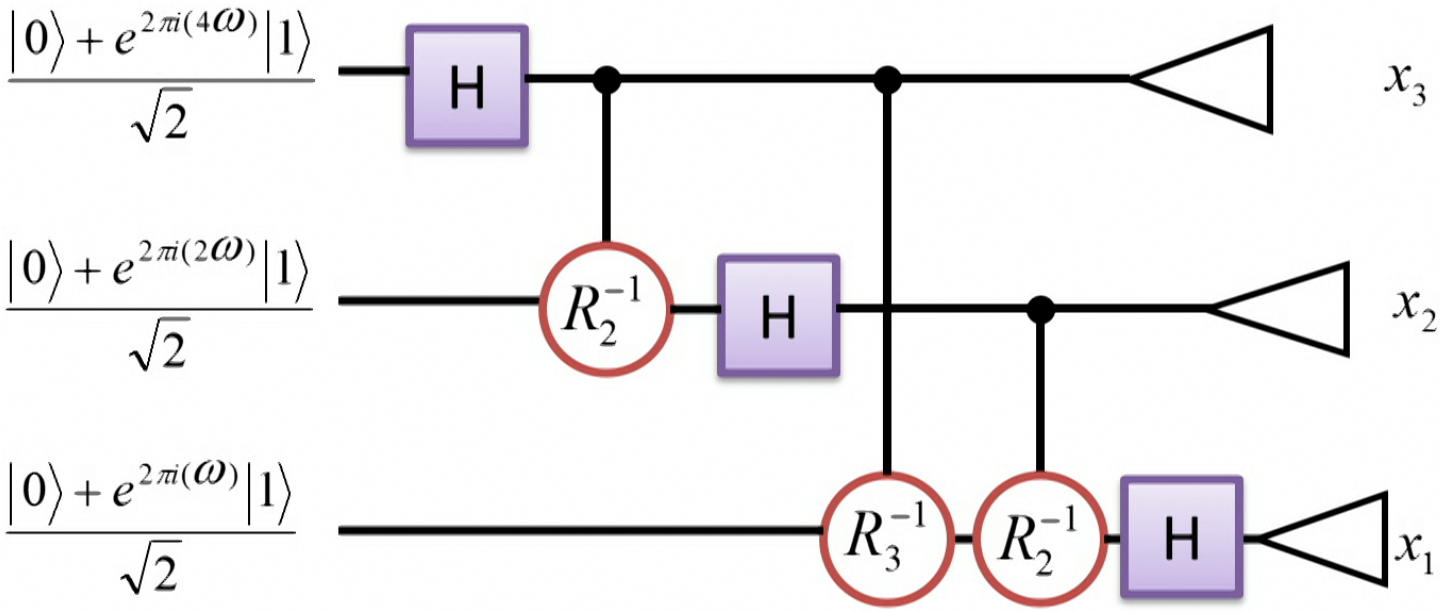
Quantum phase estimation

- For any real $\omega \in [0,1)$



Quantum phase estimation

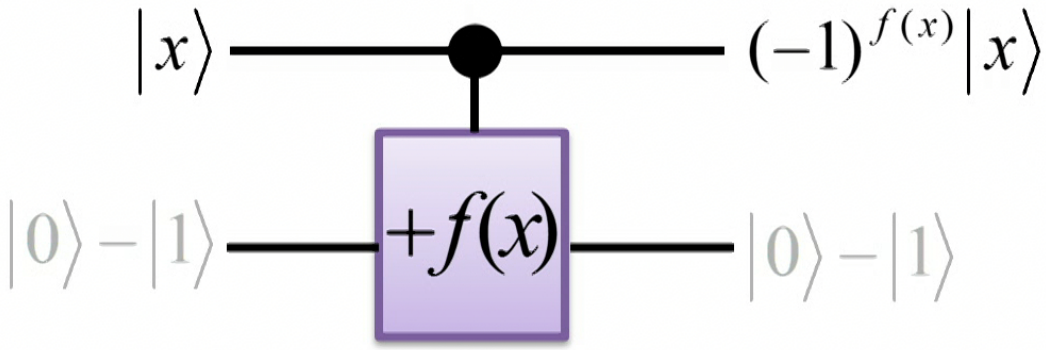
- For any real $\omega \in [0,1)$



- With high probability $\frac{4x_1 + 2x_2 + x_3}{8} \approx \omega$

Eigenvalue kick-back

- Recall the "trick":



$$|x\rangle(|0\rangle - |1\rangle) \rightarrow |x\rangle(|f(x)\rangle - |f(x) \oplus 1\rangle)$$

$$= |x\rangle(-1)^{f(x)}(|0\rangle - |1\rangle)$$

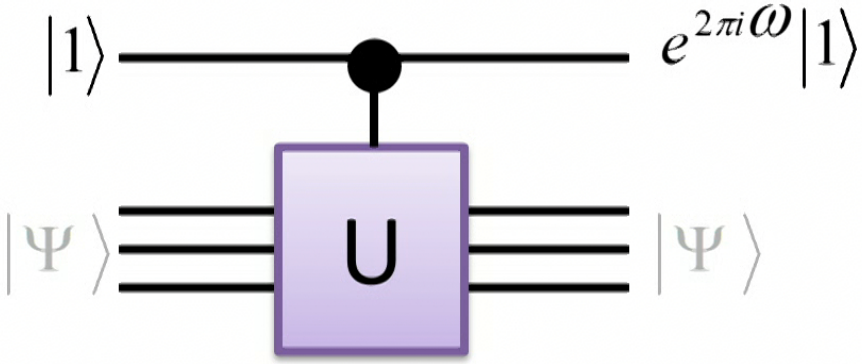
$$= (-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)$$

Eigenvalue kick-back (Kitaev)

- Consider a unitary operation **U** with eigenvalue $e^{2\pi i \omega}$ and eigenvector $|\Psi\rangle$

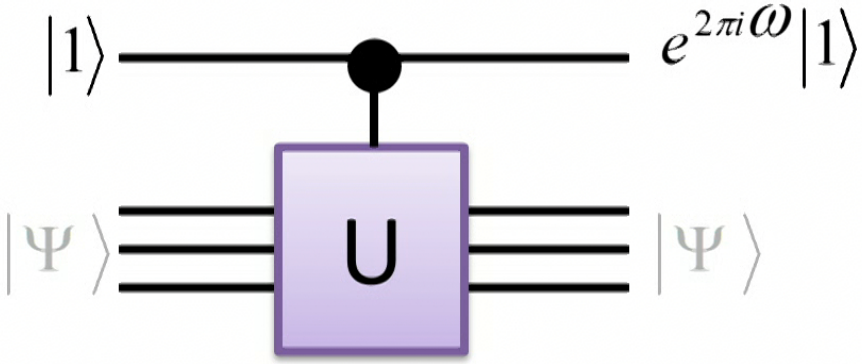
Eigenvalue kick-back (Kitaev)

- Consider a unitary operation **U** with eigenvalue $e^{2\pi i\omega}$ and eigenvector $|\Psi\rangle$



Eigenvalue kick-back (Kitaev)

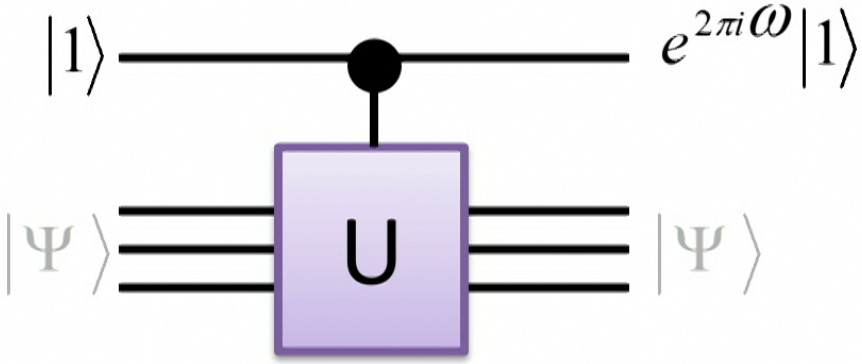
- Consider a unitary operation **U** with eigenvalue $e^{2\pi i\omega}$ and eigenvector $|\Psi\rangle$



$$|1\rangle|\Psi\rangle \rightarrow |1\rangle U|\Psi\rangle = |1\rangle e^{2\pi i\omega}|\Psi\rangle$$

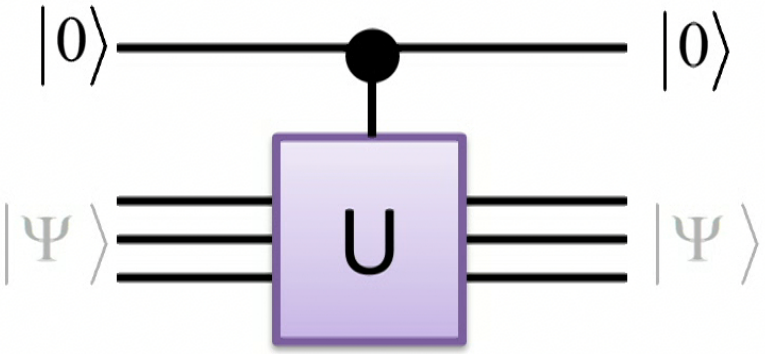
Eigenvalue kick-back (Kitaev)

- Consider a unitary operation **U** with eigenvalue $e^{2\pi i\omega}$ and eigenvector $|\Psi\rangle$

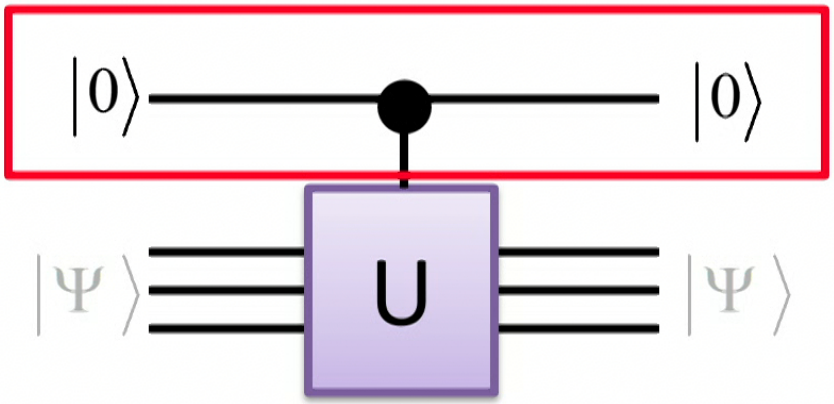


$$|1\rangle|\Psi\rangle \rightarrow |1\rangle U|\Psi\rangle = |1\rangle e^{2\pi i\omega}|\Psi\rangle$$
$$= e^{2\pi i\omega}|1\rangle|\Psi\rangle$$

Eigenvalue kick-back

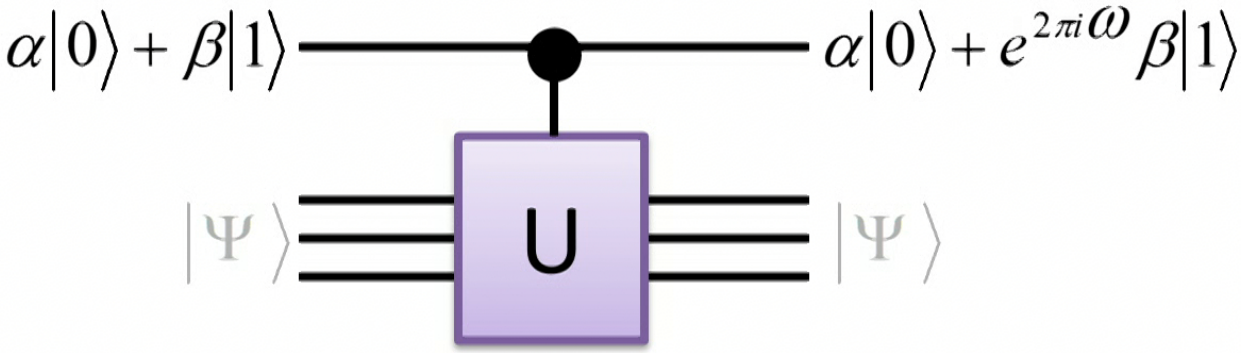


Eigenvalue kick-back



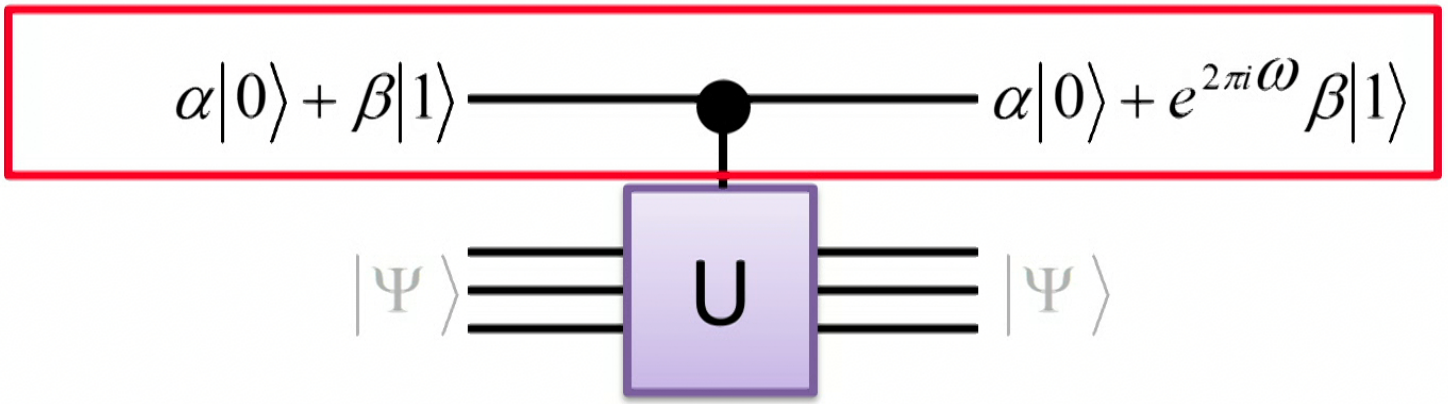
Eigenvalue kick-back

- As a relative phase, $e^{2\pi i \omega}$ becomes measurable



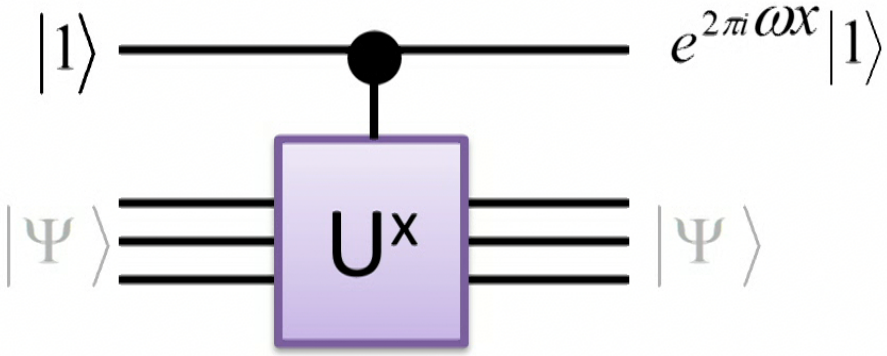
Eigenvalue kick-back

- As a relative phase, $e^{2\pi i \omega}$ becomes measurable



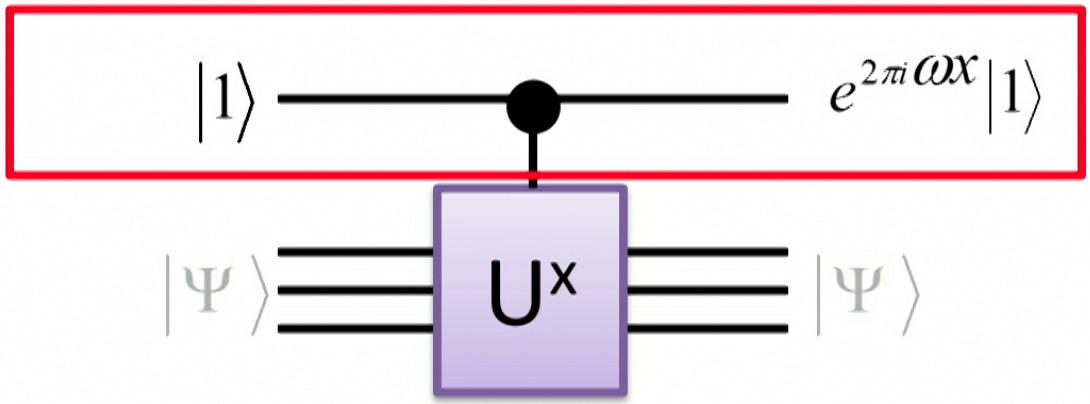
Eigenvalue kick-back

- If we exponentiate **U**, we get multiples of ω

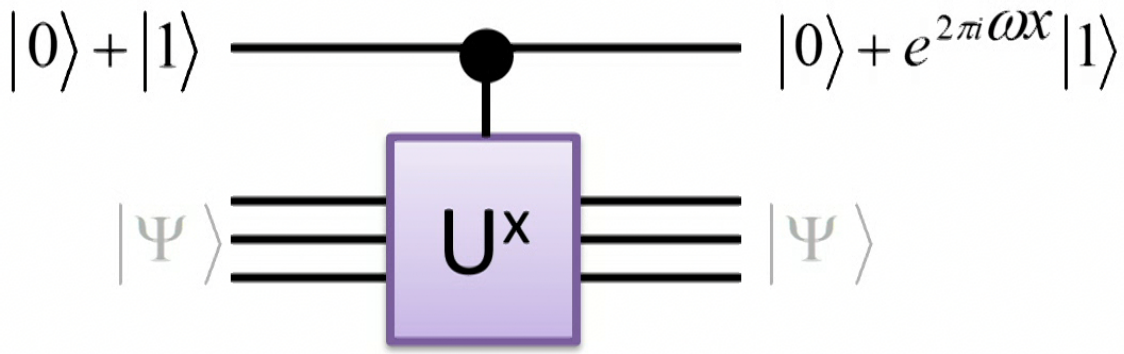


Eigenvalue kick-back

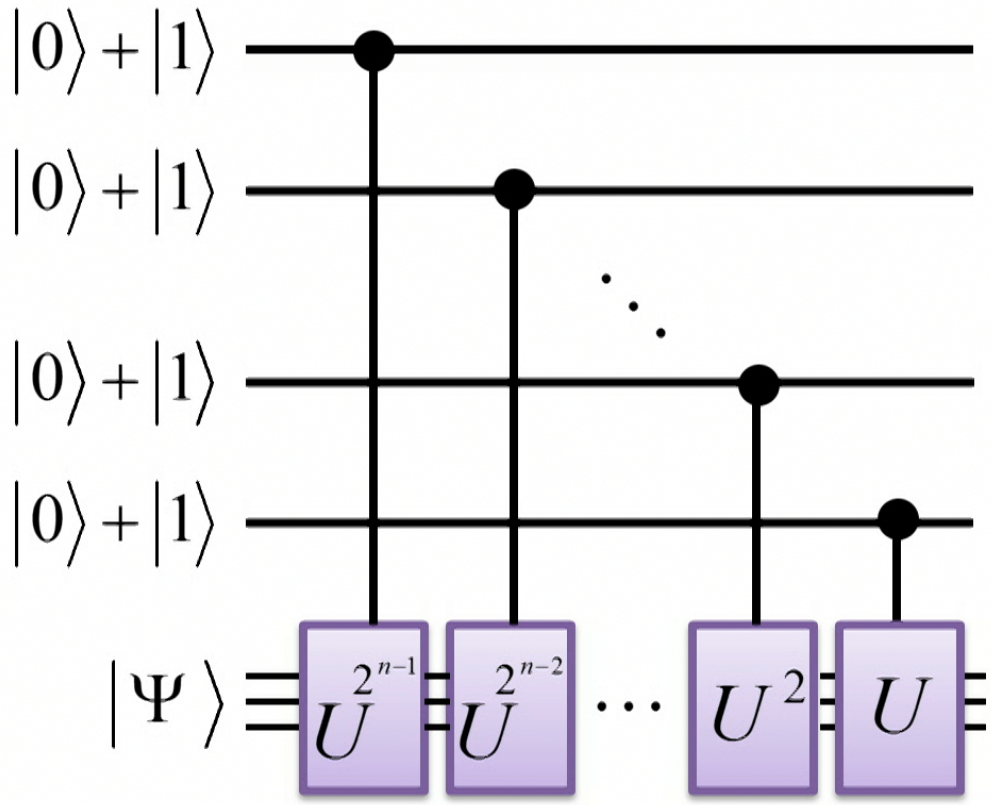
- If we exponentiate **U**, we get multiples of ω



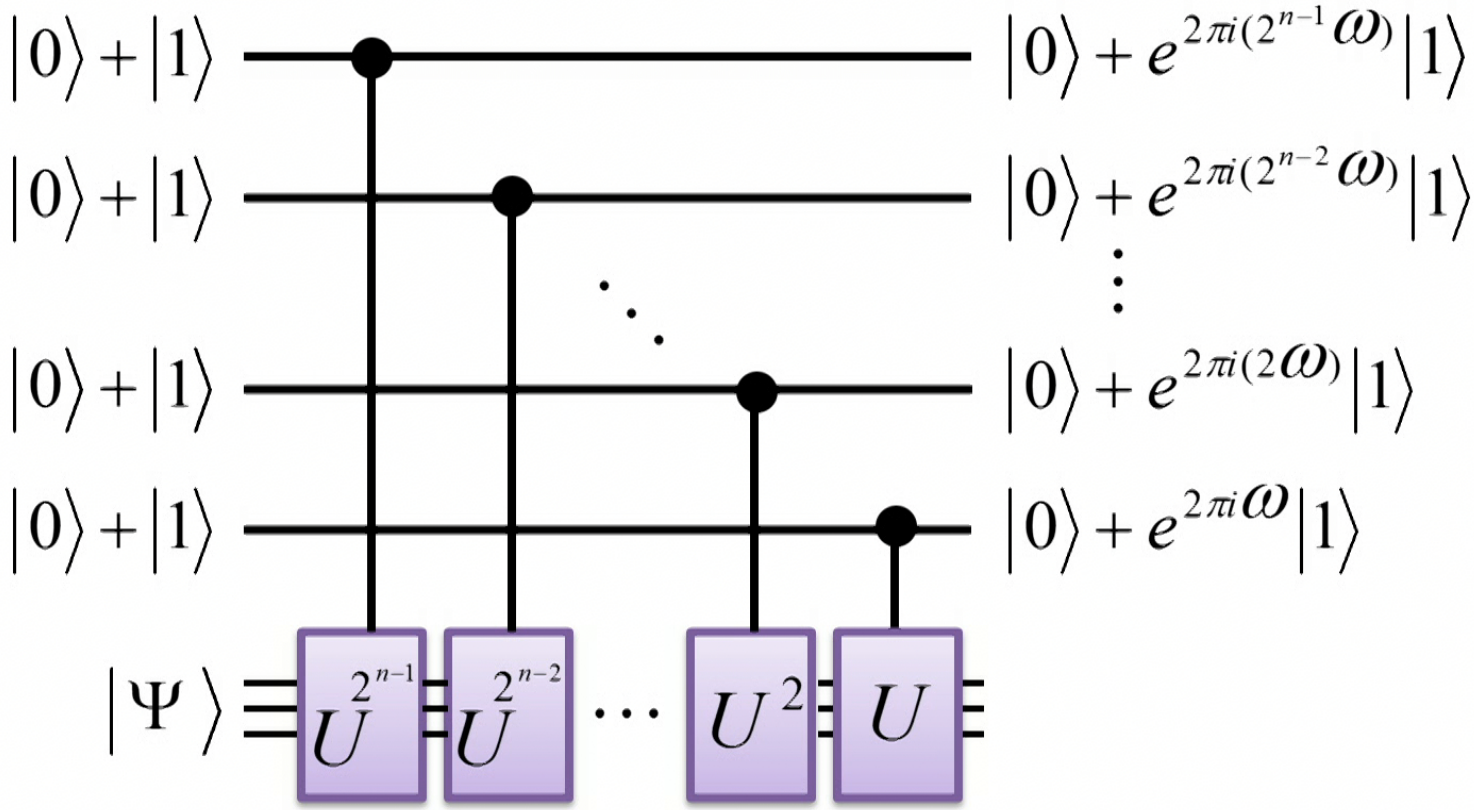
Eigenvalue kick-back



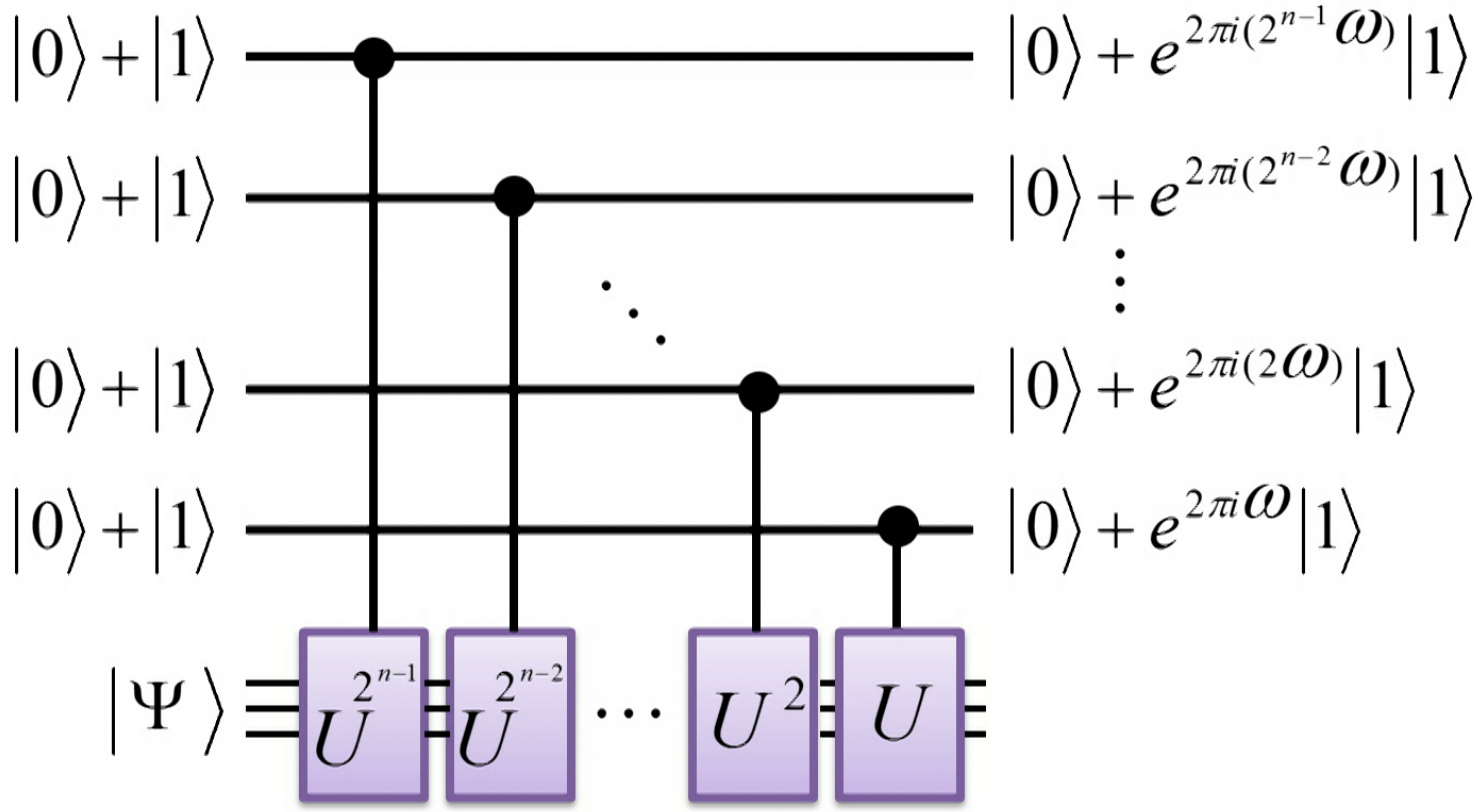
Eigenvalue kick-back



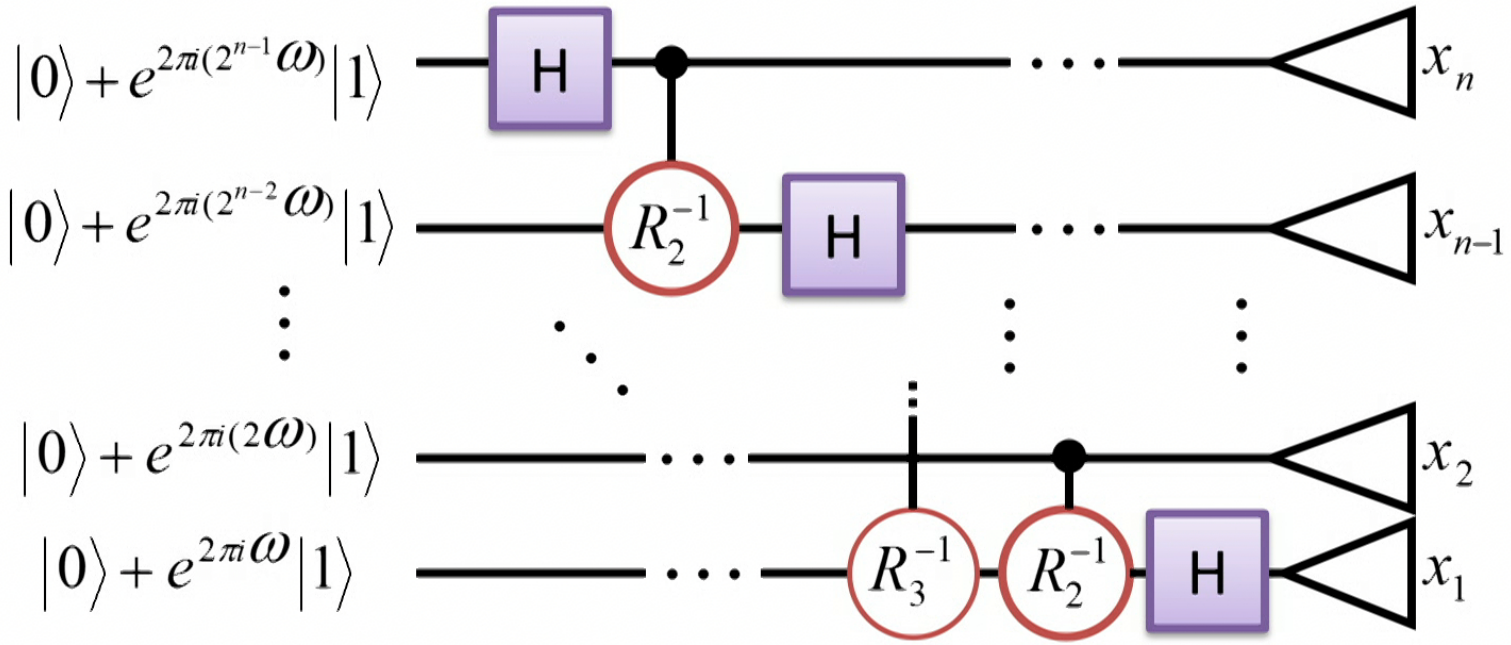
Eigenvalue kick-back



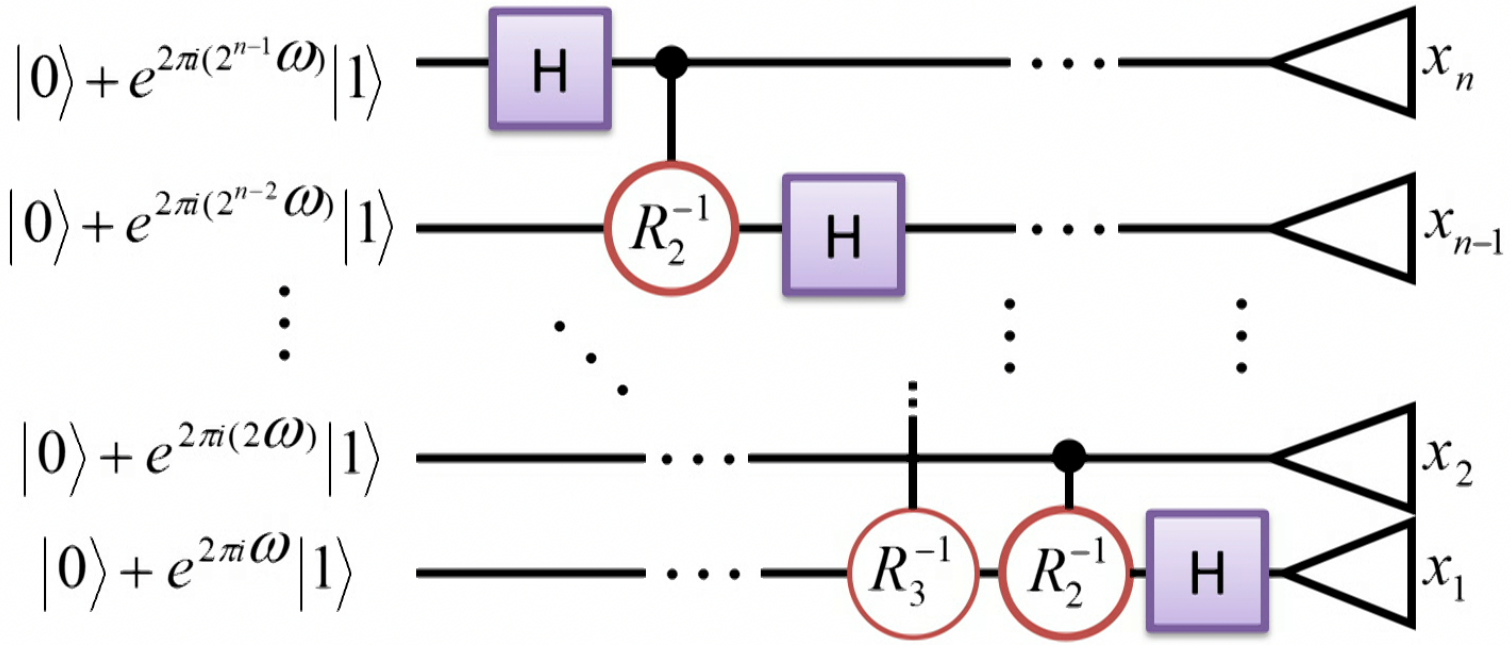
Eigenvalue kick-back



Phase estimation



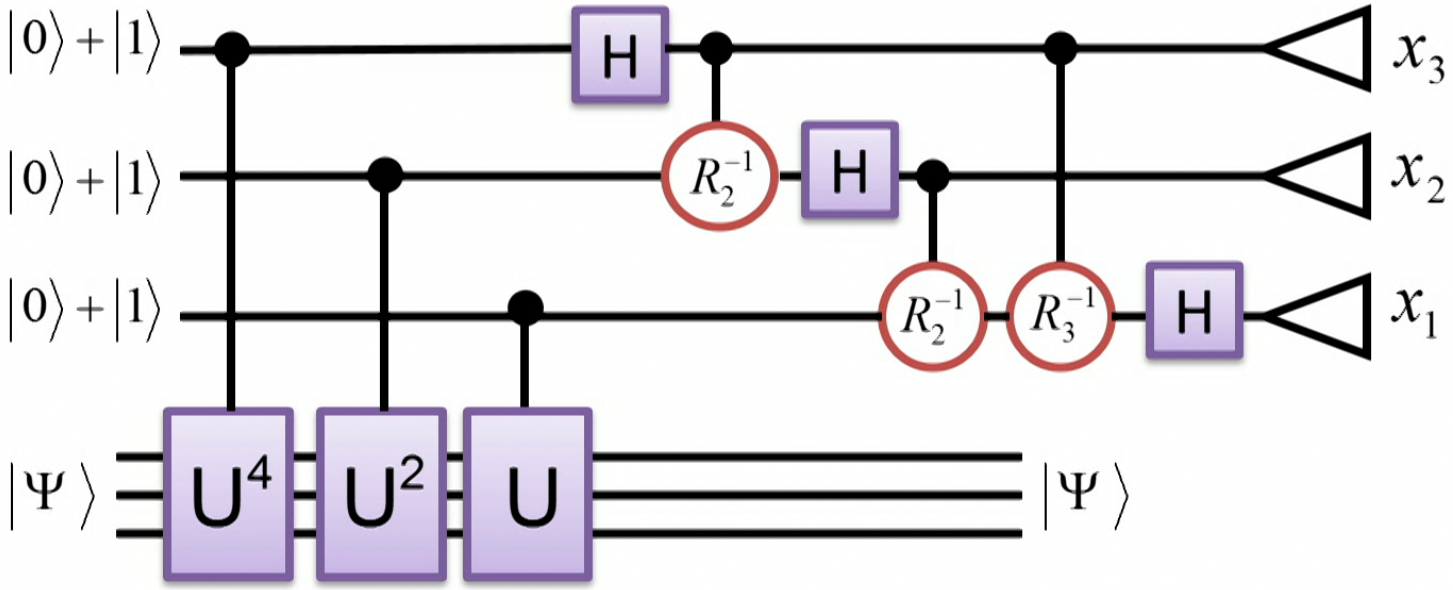
Phase estimation



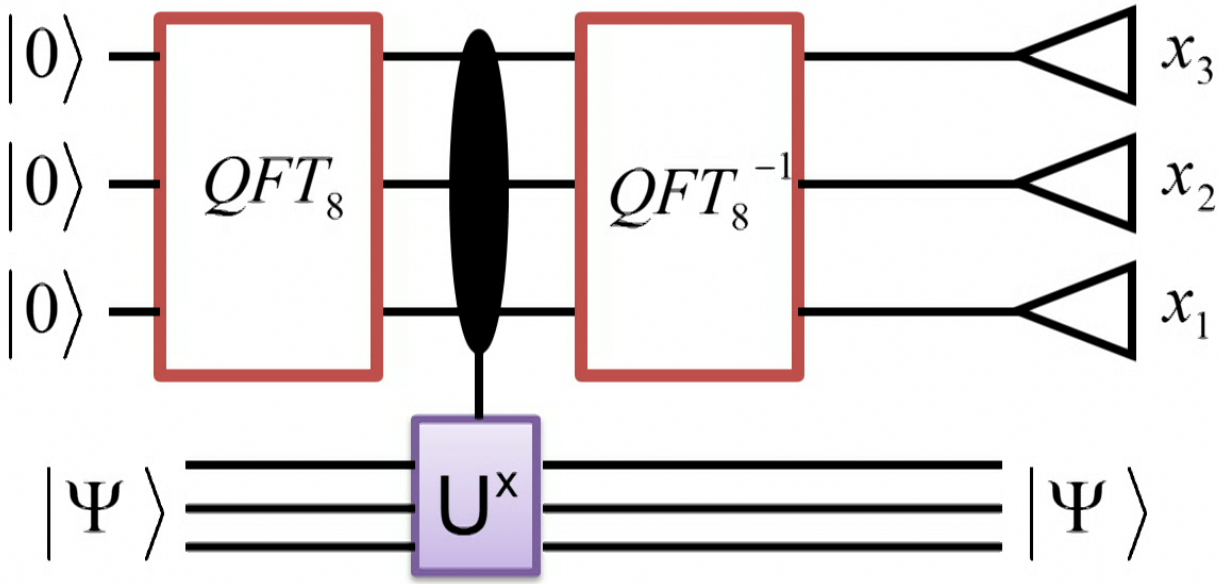
$$\frac{2^{n-1}x_1 + 2^{n-2}x_2 + \cdots x_n}{2^n} \approx \omega$$

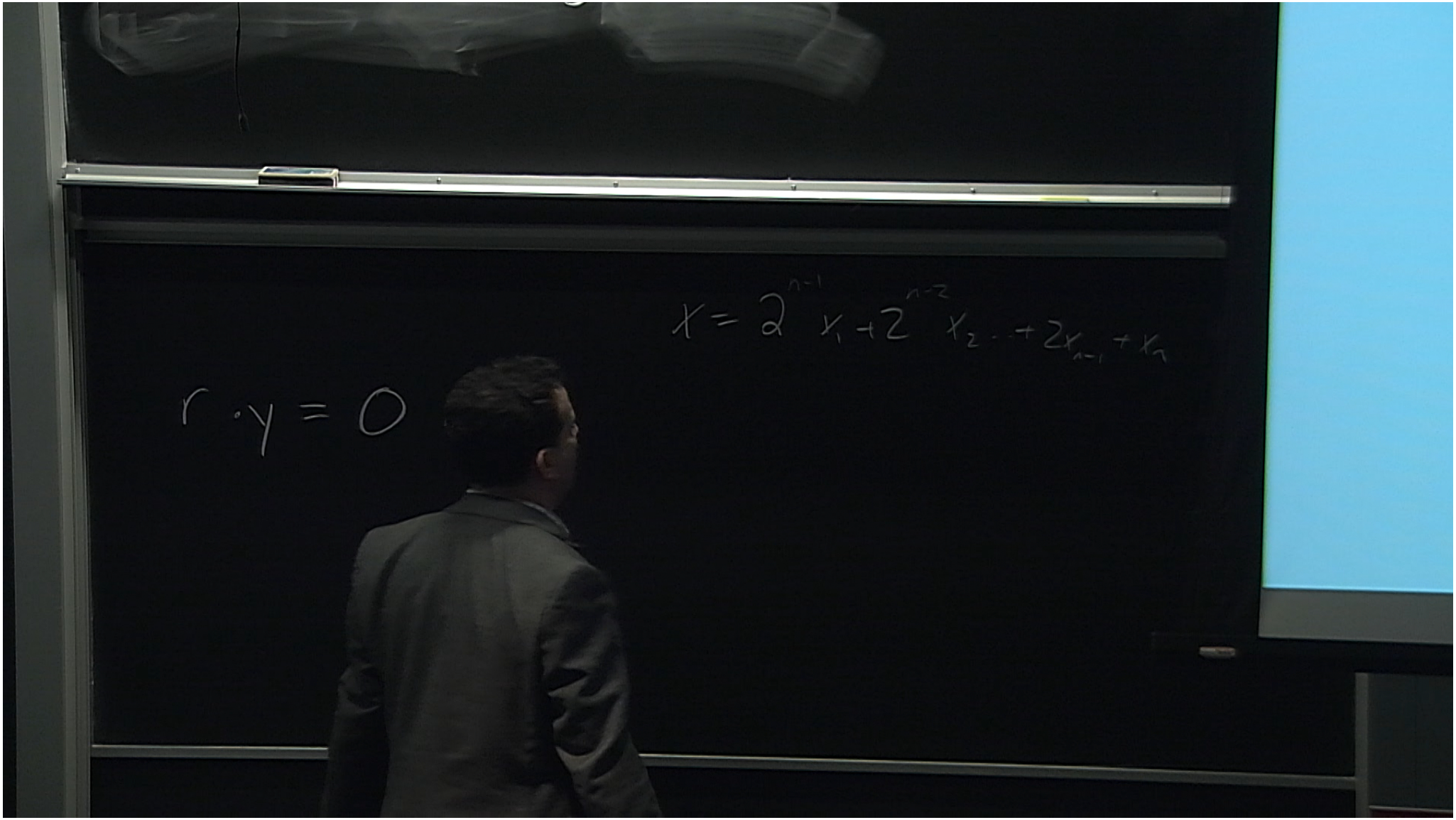
45

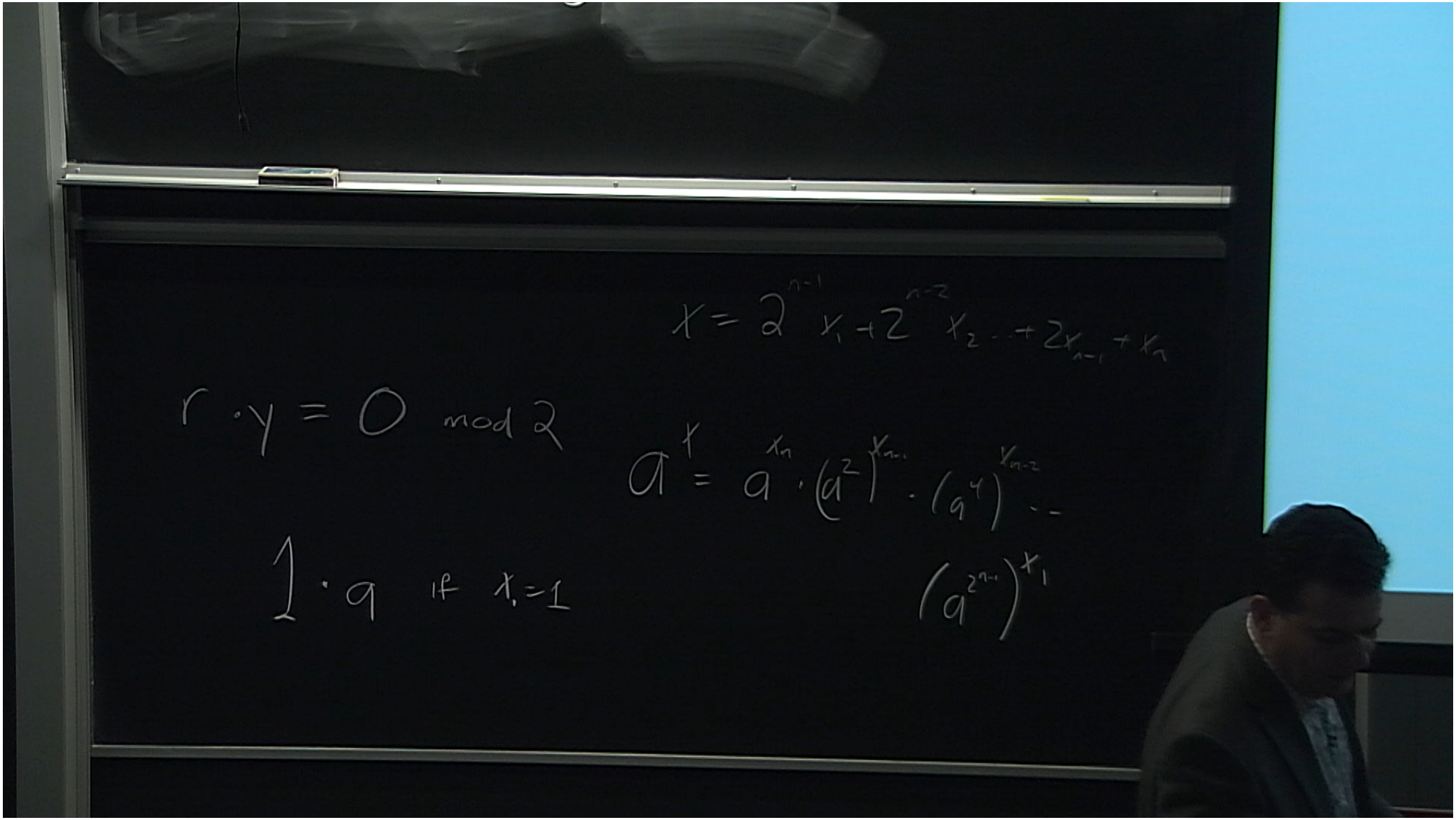
Eigenvalue estimation



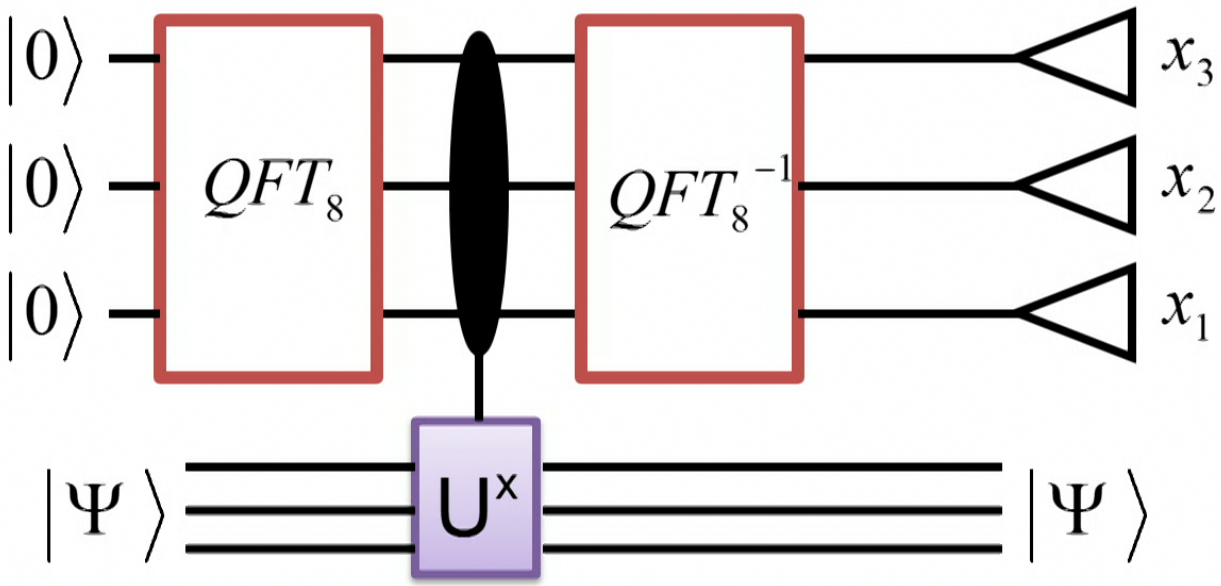
Eigenvalue estimation




$x = 2^{n-1} x_1 + 2^{n-2} x_2 \cdots + 2x_{n-1} + x_n$
$r \cdot y = 0$

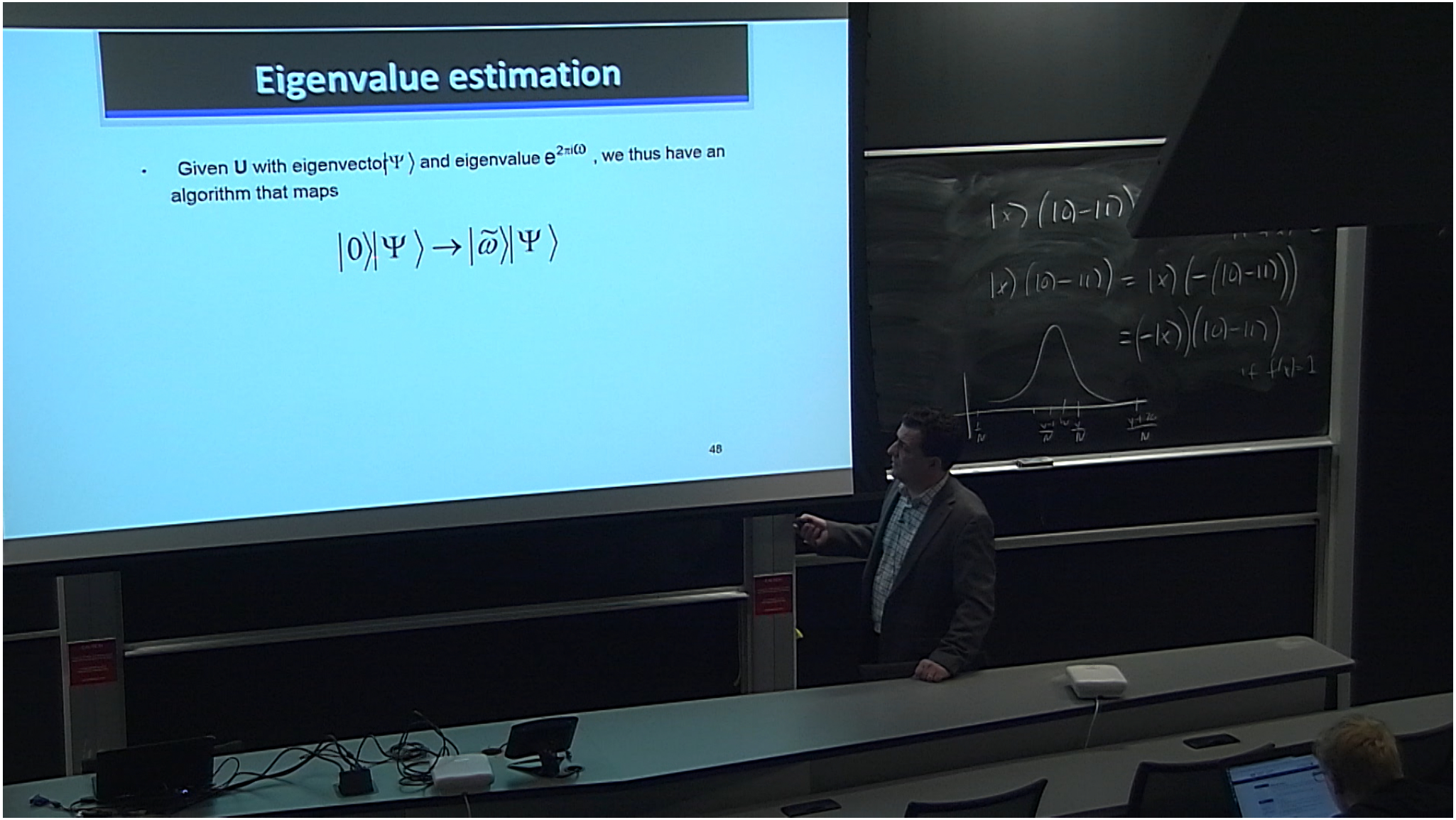

$x = 2^{n-1} x_1 + 2^{n-2} x_2 \cdots + 2x_{n-1} + x_n$
$r \cdot y = 0 \mod 2$
$a^x = a^{x_n} \cdot (a^2)^{x_{n-1}} \cdot (a^4)^{x_{n-2}} \cdots$
$1 \cdot a$ if $x_1 = 1$
$(a^{2^{n-1}})^{x_1}$

Eigenvalue estimation



Eigenvalue estimation

- Given **U** with eigenvector $|\Psi\rangle$ and eigenvalue $e^{2\pi i\omega}$, we thus have an algorithm that maps


Eigenvalue estimation
Given U with eigenvecto$|\Psi\rangle$ and eigenvalue $e^{2\pi i\omega}$, we thus have an algorithm that maps
$$|0\rangle|\Psi\rangle \rightarrow |\tilde{\omega}\rangle|\Psi\rangle$$
48

Eigenvalue estimation

- Given **U** with eigenvector $|\Psi\rangle$ and eigenvalue $e^{2\pi i\omega}$, we thus have an algorithm that maps

$$|0\rangle|\Psi\rangle \rightarrow |\tilde{\omega}\rangle|\Psi\rangle$$

Eigenvalue estimation

- Given **U** with eigenvectors $|\Psi_k\rangle$ and respective eigenvalues $e^{2\pi i\omega_k}$ we thus have an algorithm that maps

Eigenvalue estimation

- Given **U** with eigenvectors $|\Psi_k\rangle$ and respective eigenvalues $e^{2\pi i\omega_k}$ we thus have an algorithm that maps

$$|0\rangle|\Psi_k\rangle \rightarrow |\tilde{\omega}_k\rangle|\Psi_k\rangle$$

Eigenvalue estimation

- Given **U** with eigenvectors $|\Psi_k\rangle$ and respective eigenvalues $e^{2\pi i\omega_k}$ we thus have an algorithm that maps

$$|0\rangle|\Psi_k\rangle \rightarrow |\tilde{\omega}_k\rangle|\Psi_k\rangle$$

and therefore

$$|0\rangle\sum_k \alpha_k|\Psi_k\rangle = \sum_k \alpha_k|0\rangle|\Psi_k\rangle \rightarrow \sum_k \alpha_k|\tilde{\omega}_k\rangle|\Psi_k\rangle$$

Eigenvalue estimation

- Measuring the first register of

$$\sum_k \alpha_k \left| \tilde{\omega}_k \right\rangle \left| \Psi_k \right\rangle$$

Eigenvalue estimation

- Measuring the first register of

$$\sum_k \alpha_k \left| \tilde{\omega}_k \right\rangle \left| \Psi_k \right\rangle$$

is equivalent to measuring $\left| \tilde{\omega}_k \right\rangle$ with probability $\left| \alpha_k \right|^2$.

Example

- Suppose we have a group $\mathbf{G}$ and we wish to find the order of $a \in \mathbf{G}$ (i.e. the smallest positive r such that $a^r \equiv 1$).

Example

- Suppose we have a group $\mathbf{G}$ and we wish to find the order of $a \in \mathbf{G}$ (i.e. the smallest positive r such that $a^r \equiv 1$).
- If we can efficiently do arithmetic in the group, then we can realize a unitary operator $\mathbf{U_a}$ that maps $|x\rangle \rightarrow |ax\rangle$.

Example

- Suppose we have a group **G** and we wish to find the order of $a \in \mathbf{G}$ (i.e. the smallest positive r such that $a^r \equiv 1$).
- If we can efficiently do arithmetic in the group, then we can realize a unitary operator $\mathbf{U_a}$ that maps $|x\rangle \rightarrow |ax\rangle$.
- Notice that

$$U_a^{\,r} = U_{a^r} = I$$

Example

- Suppose we have a group **G** and we wish to find the order of $a \in \mathbf{G}$ (i.e. the smallest positive r such that $a^r \equiv 1$).
- If we can efficiently do arithmetic in the group, then we can realize a unitary operator $\mathbf{U_a}$ that maps $|x\rangle \rightarrow |ax\rangle$.
- Notice that

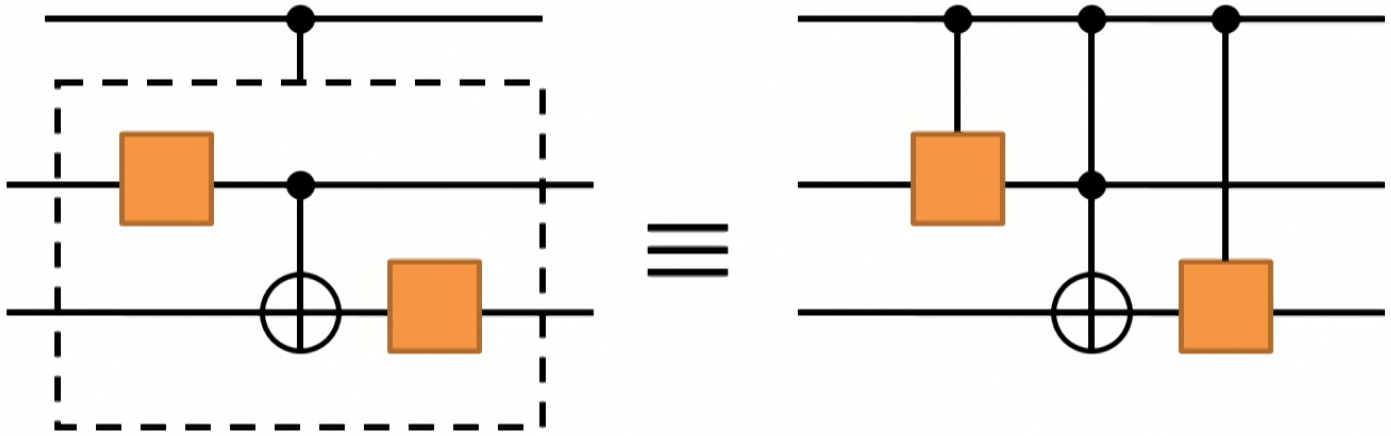
$$U_a{}^r = U_{a^r} = I$$

- This means that the eigenvalues of $\mathbf{U_a}$ are of the form $e^{2\pi i \frac{k}{r}}$ where k is an integer.

How do we implement c-U ?

Replace every gate G in the circuit with a c-G.

For example,

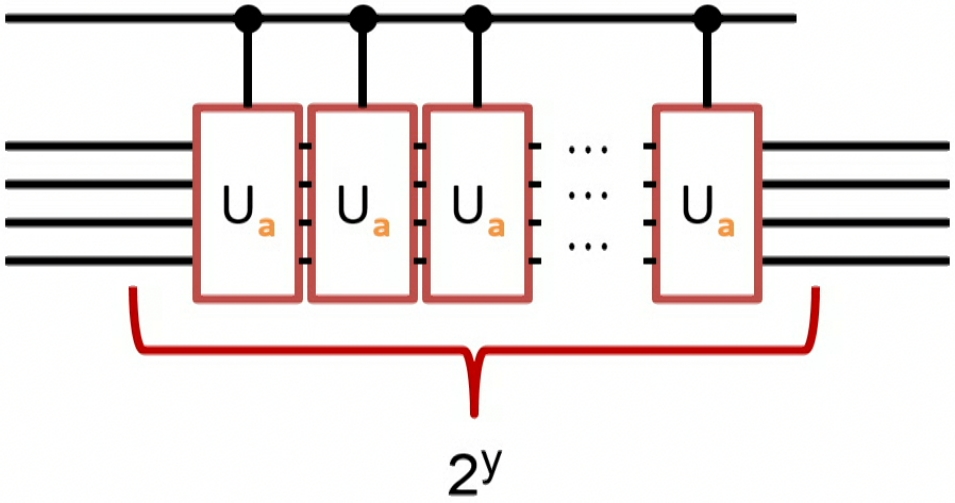


Inefficient exponentiation

We can effect a relative phase shift of $e^{i 2\pi \left(2^y \frac{k}{r} \right)}$

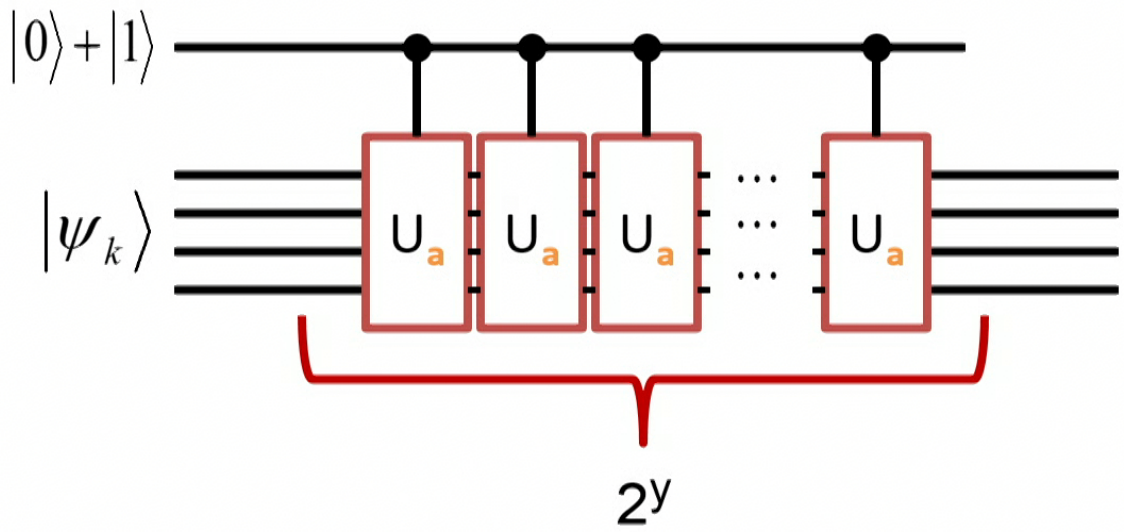
Inefficient exponentiation

We can effect a relative phase shift of $e^{i 2\pi \left(2^y \frac{k}{r} \right)}$



Inefficient exponentiation

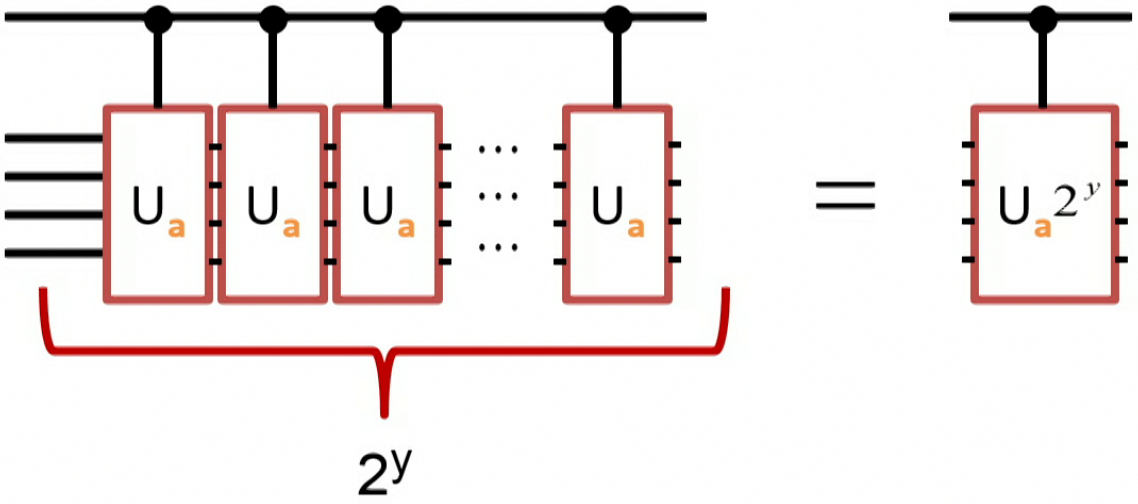
We can effect a relative phase shift of $e^{i 2\pi \left(2^y \frac{k}{r} \right)}$



Efficient exponentiation

But we can also do it **efficiently** by noticing that

$$U_a^{2^y} = U_{a^{2^y}}$$



Quantum factoring

- The security of many public key cryptosystems used in industry today relies on the difficulty of factoring large numbers into smaller factors.

- Factoring the integer *N* into smaller factors can be reduced to the following task:

Quantum factoring

- The security of many public key cryptosystems used in industry today relies on the difficulty of factoring large numbers into smaller factors.

- Factoring the integer *N* into smaller factors can be reduced to the following task:

Given integer a, find the smallest positive integer r so that $a^r \equiv 1 \bmod N$

(Aside: how does factoring reduce to order-finding ?)

- The most common approach for factoring integers is the difference of squares technique:
  - "Randomly" find two integers *x* and *y* satisfying

$$x^2 = y^2 \bmod N$$

(Aside: how does factoring reduce to order-finding ?)

- The most common approach for factoring integers is the difference of squares technique:
  - "Randomly" find two integers *x* and *y* satisfying

$$x^2 = y^2 \bmod N$$

  - So *N* divides $x^2 - y^2 = (x - y)(x + y)$

(Aside: how does factoring reduce to order-finding ?)

- The most common approach for factoring integers is the difference of squares technique:
  - "Randomly" find two integers *x* and *y* satisfying

$$x^2 = y^2 \bmod N$$

  - So *N* divides $x^2 - y^2 = (x - y)(x + y)$
  - Hope that $\gcd(N, x - y)$ is non-trivial

(Aside: how does factoring reduce to order-finding ?)

- The most common approach for factoring integers is the difference of squares technique:
  - "Randomly" find two integers *x* and *y* satisfying

$$x^2 = y^2 \bmod N$$

  - So *N* divides $x^2 - y^2 = (x - y)(x + y)$
  - Hope that $\gcd(N, x - y)$ is non-trivial

- If *r* is even, then let

$$x = a^{r/2} \bmod N$$

(Aside: how does factoring reduce to order-finding ?)

- The most common approach for factoring integers is the difference of squares technique:
  - "Randomly" find two integers *x* and *y* satisfying

$$x^2 = y^2 \bmod N$$

  - So *N* divides $x^2 - y^2 = (x - y)(x + y)$
  - Hope that $\gcd(N, x - y)$ is non-trivial

- If *r* is even, then let

$$x = a^{r/2} \bmod N$$

so that

$$x^2 = 1^2 \bmod N$$

Quantum factoring

Since we know how to efficiently multiply by a mod N, we can efficiently implement

$$U_a|x\rangle = |ax\rangle$$

Note that

$$U_a^r|x\rangle = |a^r x\rangle = |x\rangle$$

Quantum factoring

Since we know how to efficiently multiply by a mod N, we can efficiently implement

$$U_a|x\rangle = |ax\rangle$$

Note that

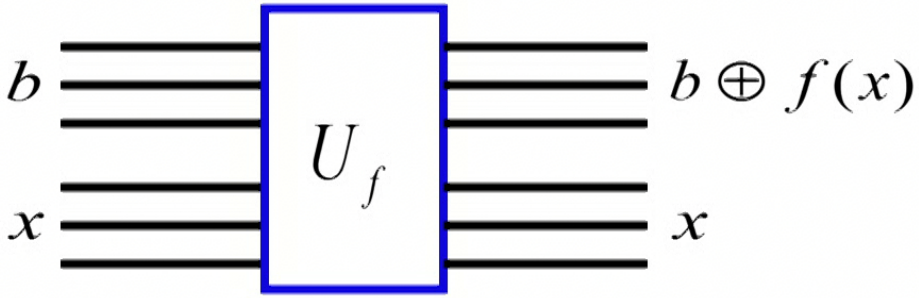
$$U_a^r|x\rangle = |a^r x\rangle = |x\rangle$$

i.e.

$$U_a^r = \mathrm{I}$$

(Aside : more on reversible computing)

If we know how to efficiently compute f and f^{-1} then we can efficiently and reversibly map



(Aside : more on reversible computing)

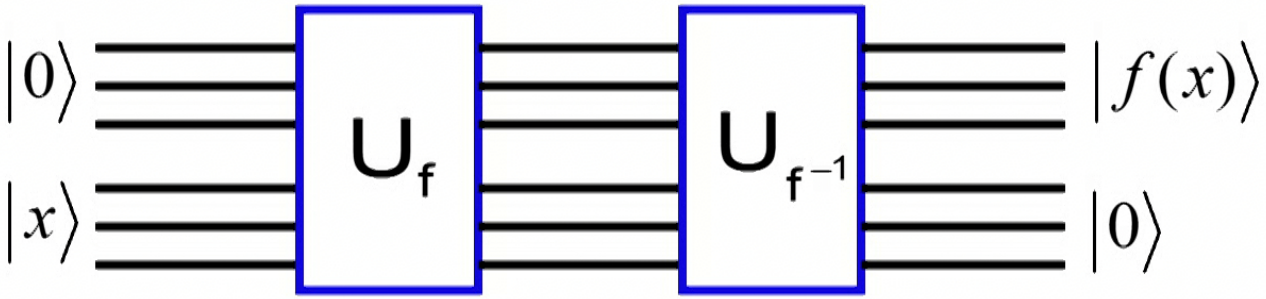
And therefore we can efficiently map

$$|x\rangle \mapsto |f(x)\rangle$$

(Aside : more on reversible computing)

And therefore we can efficiently map

$$|x\rangle \mapsto |f(x)\rangle$$



Interesting eigenvalues

If $U_a^r = I$ then the eigenvalues of U_a are of the form

$$e^{2\pi i \frac{k}{r}}$$

Interesting eigenvalues

If $U_a^r = I$ then the eigenvalues of U_a are of the form

$$e^{2\pi i \frac{k}{r}}$$

$$\Rightarrow \quad U_a \left| \psi_k \right\rangle = \mathrm{e}^{\mathrm{i}2\pi\frac{\mathrm{k}}{r}} \left| \psi_k \right\rangle$$

Checking the eigenvalues

$$U_a \left|\psi_k\right\rangle = \sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}} U_a \left|a^j\right\rangle$$

$$= \sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}} \left|a^{j+1}\right\rangle = e^{i2\pi\frac{k}{r}} \left(\sum_{j=1}^{r} e^{-i2\pi j\frac{k}{r}} \left|a^j\right\rangle \right)$$

$$= e^{i2\pi\frac{k}{r}} \left(\sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}} \left|a^j\right\rangle \right)$$

Finding r

For most integers k, a good estimate of $\frac{k}{r}$ (with error at most $\frac{1}{2r^2}$) allows us to determine r (even if we don't know k).

(using continued fractions)

Complexity comparison

•The best rigorous classical algorithms use

$$e^{O(\sqrt{\log(N)\log\log(N)})}$$

operations

Complexity comparison

•The best rigorous classical algorithms use

$$e^{O(\sqrt{\log(N)\log\log(N)})}$$

operations

•The best heuristic classical algorithms use

$$e^{O((\log(N)^{\frac{1}{3}}\log\log(N)^{\frac{2}{3}})}$$

operations

63

Complexity comparison

•The best rigorous classical algorithms use

$$e^{O(\sqrt{\log(N)\log\log(N)})}$$

operations

•The best heuristic classical algorithms use

$$e^{O((\log(N)^{\frac{1}{3}}\log\log(N)^{\frac{2}{3}})}$$

operations

•The quantum algorithm uses poly(log(N))

$$= e^{O(\log\log(N))}$$

operations

63

Overview

- Quantum searching
- Quantum counting
- Searching when you don’t know the number of elements

Complexity comparison

•The best rigorous classical algorithms use

$$e^{O(\sqrt{\log(N)\log\log(N)})}$$

operations

•The best heuristic classical algorithms use

$$e^{O((\log(N)^{\frac{1}{3}}\log\log(N)^{\frac{2}{3}})}$$

operations

•The quantum algorithm uses poly(log(N))

$$= e^{O(\log\log(N))}$$

operations

63

Overview

- Quantum searching
- Quantum counting
- Searching when you don’t know the number of elements


QUANTUM SEARCHING

Searching problem

Consider

$$f : \{0,1\}^n \rightarrow \{0,1\}$$

Searching problem

Consider

$$f : \{0,1\}^n \rightarrow \{0,1\}$$

Given

$$U_f : |x\rangle \mapsto (-1)^{f(x)} |x\rangle$$

Searching problem

Consider

$$f : \{0,1\}^n \rightarrow \{0,1\}$$

Given

$$U_f : |x\rangle \mapsto (-1)^{f(x)} |x\rangle$$

Find an x satisfying $f(x) = 1$

Application

Consider a 3-SAT formula

$$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_M$$

Application

Consider a 3-SAT formula

$$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_M$$

$$C_j = (y_{j,1} \vee y_{j,2} \vee y_{j,2})$$

$$y_{j,k} \in \{x_1, x_2, \cdots, x_n, \bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n\}$$

Application

Consider a 3-SAT formula

$$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_M$$

$$C_j = (y_{j,1} \vee y_{j,2} \vee y_{j,2})$$

$$y_{j,k} \in \{x_1, x_2, \cdots, x_n, \bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n\}$$

For a given assignment $\mathbf{x} = \mathbf{x}_1 \mathbf{x}_2 \cdots \mathbf{x}_n$

$$f_\Phi(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ satisfies } \Phi \\ 0 & \text{otherwise} \end{cases}$$

Some ideas

For simplicity, let's start by assuming that $f(x) = 1$ has exactly one solution, $x = w$.

Some ideas

For simplicity, let's start by assuming that $f(x) = 1$ has exactly one solution, $x = w$.

IDEA:

Prepare

$$\sum_x \frac{1}{\sqrt{2^n}}|x\rangle = \frac{1}{\sqrt{2^n}}|w\rangle + \left(\sum_{x \neq w} \frac{1}{\sqrt{2^n}}|x\rangle\right)$$

Some ideas

For simplicity, let's start by assuming that $f(x) = 1$ has exactly one solution, $x = w$.

IDEA:

Prepare

$$\sum_x \frac{1}{\sqrt{2^n}}|x\rangle = \frac{1}{\sqrt{2^n}}|w\rangle + \left(\sum_{x \neq w} \frac{1}{\sqrt{2^n}}|x\rangle\right)$$

Keep this

Some ideas

For simplicity, let's start by assuming that $f(x) = 1$ has exactly one solution, $x = w$.

IDEA:

Prepare

$$\sum_x \frac{1}{\sqrt{2^n}}|x\rangle = \frac{1}{\sqrt{2^n}}|w\rangle + \left(\sum_{x \neq w} \frac{1}{\sqrt{2^n}}|x\rangle\right)$$

Keep this

"Re-scramble" this

Some ideas

For simplicity, let's start by assuming that $f(x) = 1$ has exactly one solution, $x = w$.

IDEA:

Prepare

$$\sum_x \frac{1}{\sqrt{2^n}}|x\rangle = \frac{1}{\sqrt{2^n}}|w\rangle + \left(\sum_{x \neq w} \frac{1}{\sqrt{2^n}}|x\rangle\right)$$

Keep this

"Re-scramble" this

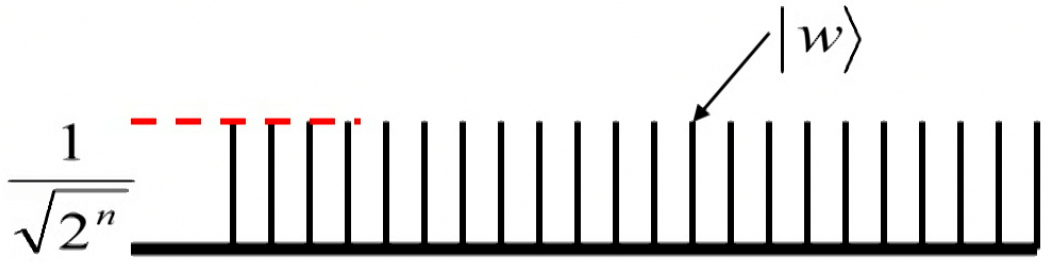
Repeat roughly $\sqrt{2^n}$ times.

! Must be done with legal quantum operation

Grover's idea:

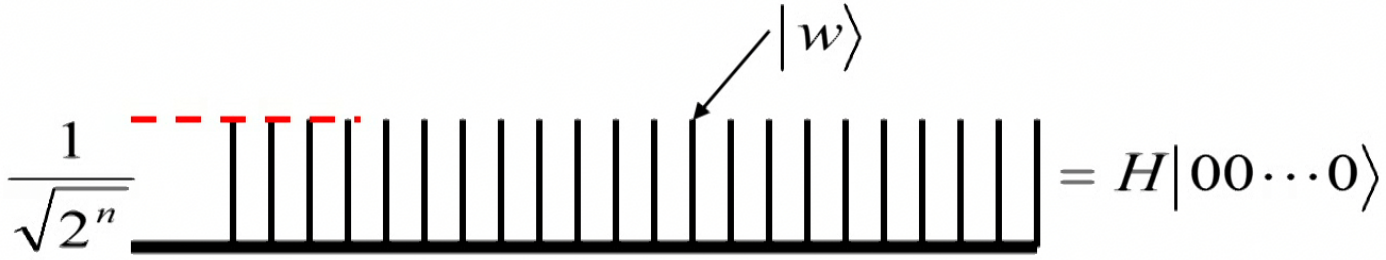
! Must be done with legal quantum operation

Grover's idea:



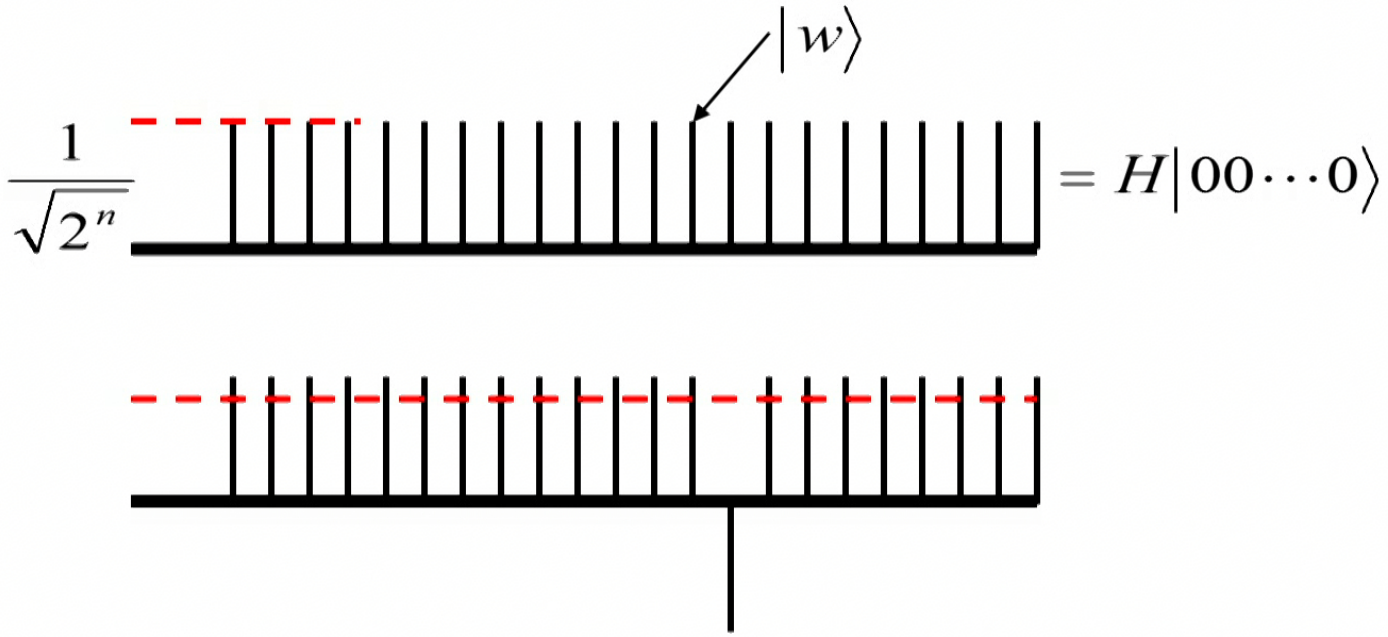
! Must be done with legal quantum operation

Grover's idea:



! Must be done with legal quantum operation

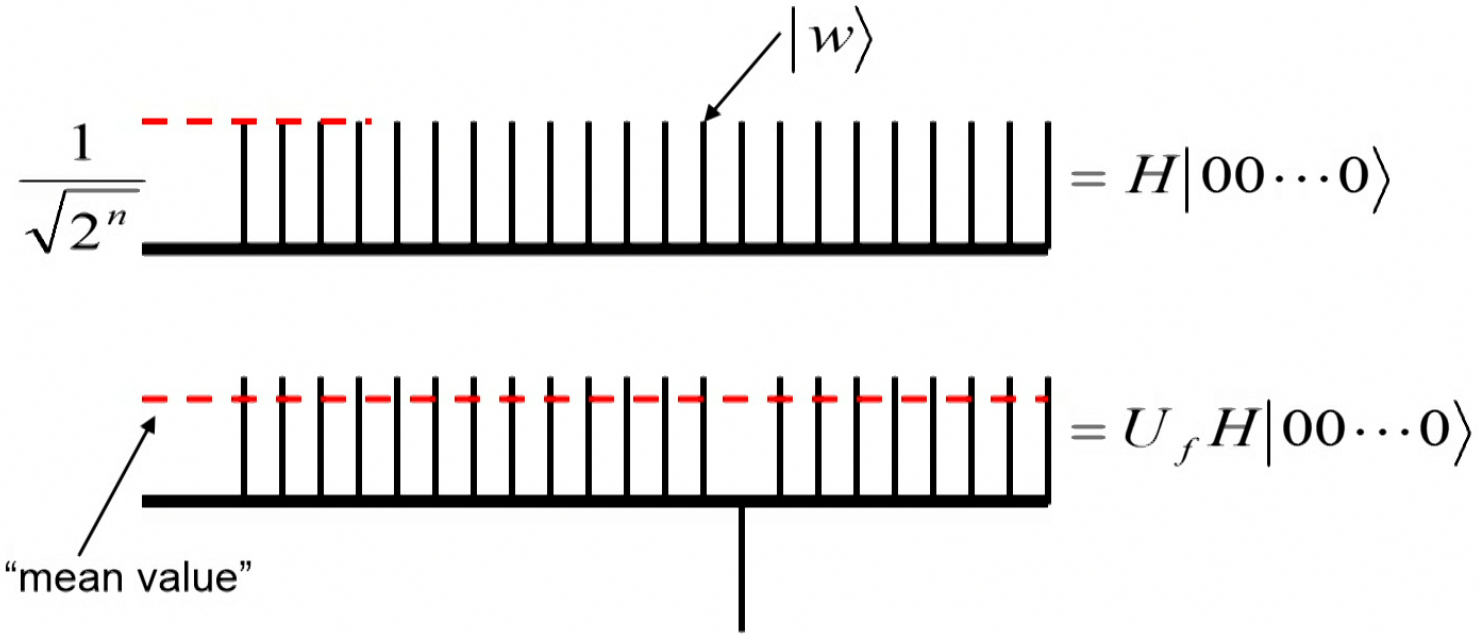
Grover's idea:



69

! Must be done with legal quantum operation

Grover's idea:

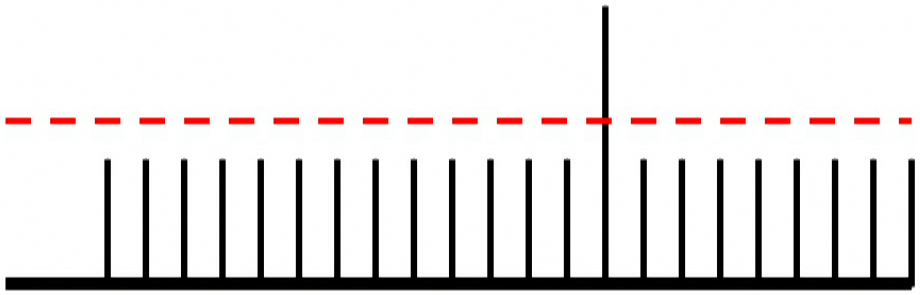


! Must be done with legal quantum operation

"invert about the mean"

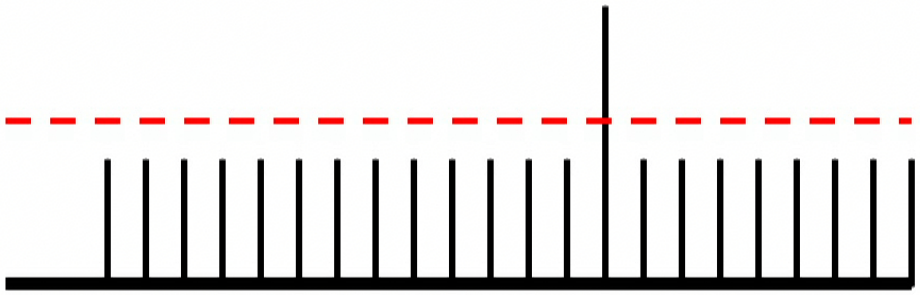
! Must be done with legal quantum operation

"invert about the mean"



! Must be done with legal quantum operation

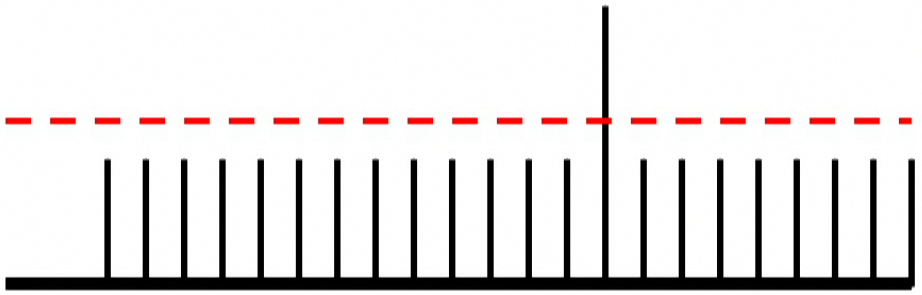
"invert about the mean"



$$= \left(-HU_0H\right)U_f H\left|00\cdots0\right\rangle$$

! Must be done with legal quantum operation

"invert about the mean"

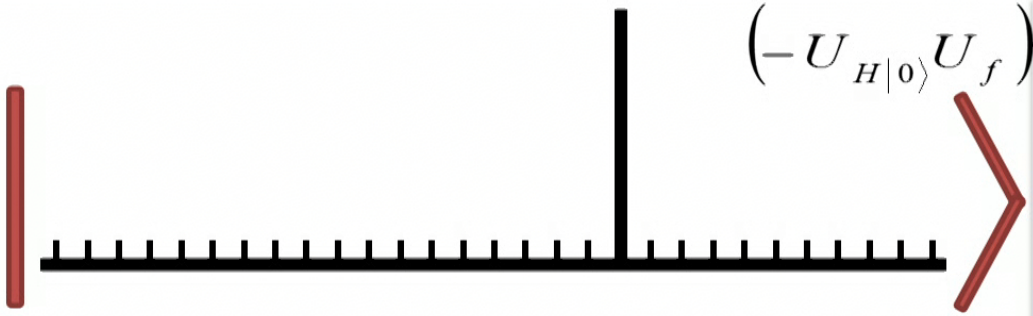


$$= \left(-HU_0H\right)U_f H\left|00\cdots0\right\rangle$$

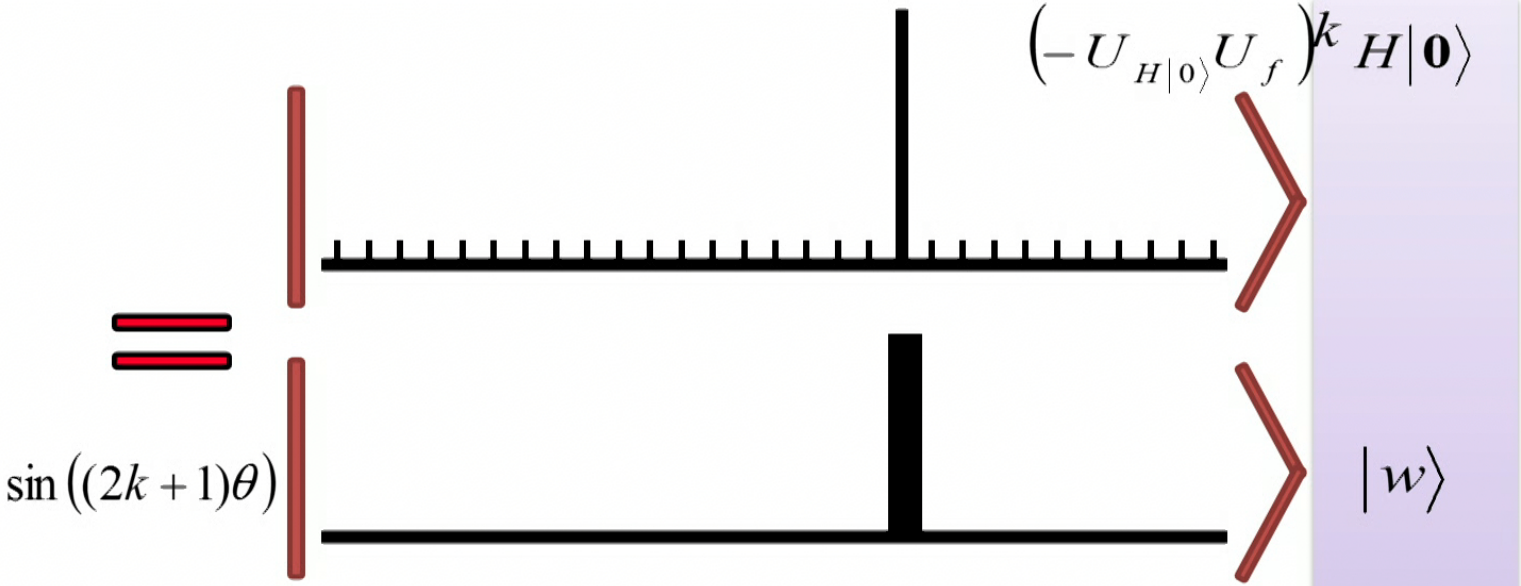
$$= -U_{H\left|0\right\rangle}U_f H\left|00\cdots0\right\rangle$$

*formula found by BBHT

$$\left(-U_{H|0\rangle}U_f\right)^k H|\mathbf{0}\rangle$$

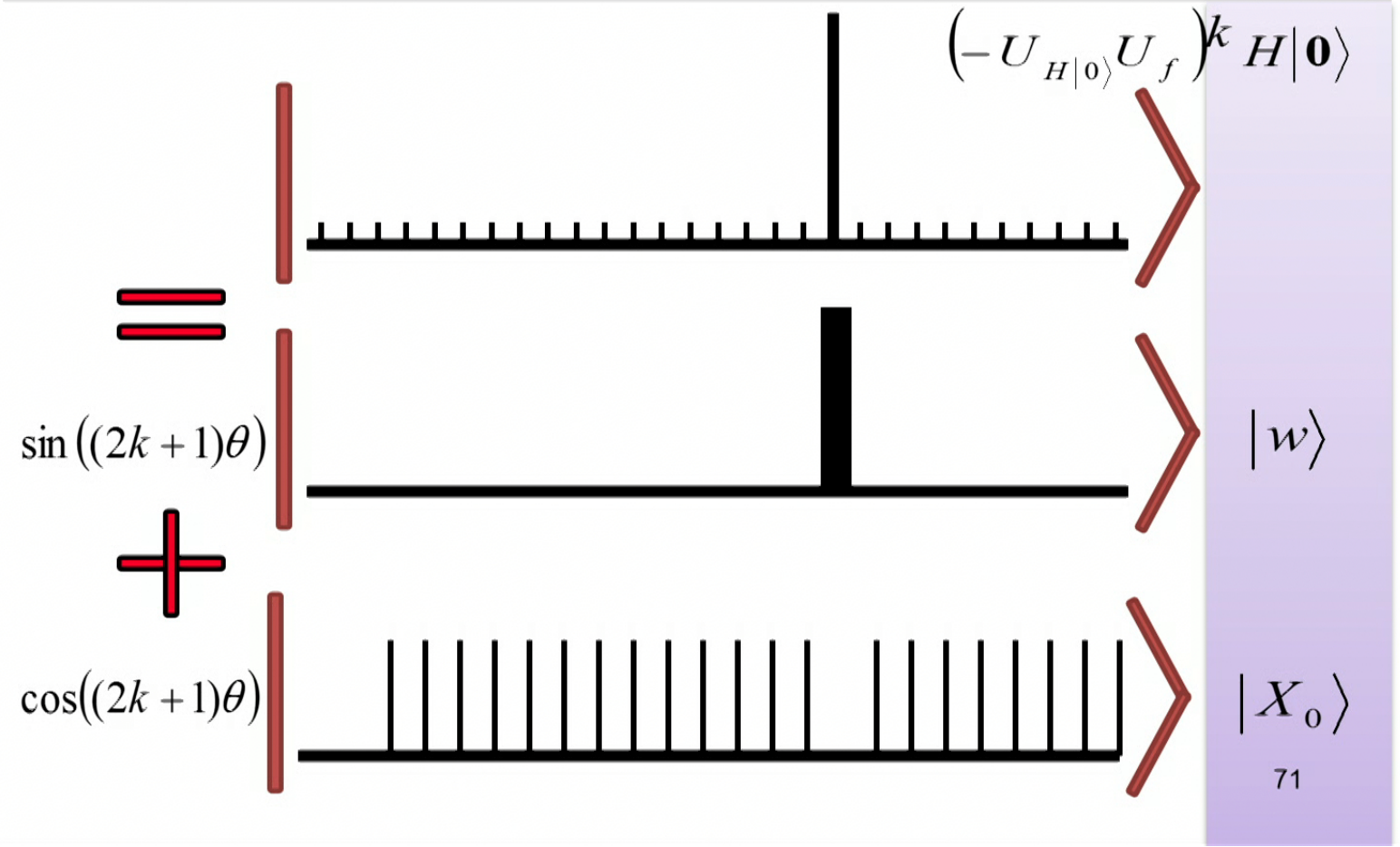


*formula found by BBHT



71

*formula found by BBHT



71

Selecting parameters

So we need

$$\sin\left((2k+1)\theta\right) \approx 1$$

$$k \approx \frac{\pi}{4\theta} - \frac{1}{2} \approx \frac{\pi\sqrt{2^n}}{4}$$

Selecting parameters

So we need

$$\sin\left((2k+1)\theta\right) \approx 1$$

$$k \approx \frac{\pi}{4\theta} - \frac{1}{2} \approx \frac{\pi\sqrt{2^n}}{4}$$

Selecting parameters

So we need

$$\sin\left((2k+1)\theta\right) \approx 1$$

$$k \approx \frac{\pi}{4\theta} - \frac{1}{2} \approx \frac{\pi\sqrt{2^n}}{4}$$

Square root speed-up!

What if we don't know k ?

Generalization: Amplitude Amplification
(BBHT,BH,BHT,G,BHMT,…)

Consider functions with t solutions

$$X_1 = f^{-1}(1) \qquad X_0 = f^{-1}(0) \qquad t = |X_1|$$

Generalization: Amplitude Amplification
(BBHT,BH,BHT,G,BHMT,…)

Consider functions with t solutions

$$X_1 = f^{-1}(1) \qquad X_0 = f^{-1}(0) \qquad t = |X_1|$$

Consider any algorithm that works with non-zero probability $p = \sin^2(\theta)$

Generalization: Amplitude Amplification
(BBHT,BH,BHT,G,BHMT,…)

Consider functions with t solutions

$$X_1 = f^{-1}(1) \qquad X_0 = f^{-1}(0) \qquad t = |X_1|$$

Consider any algorithm that works with non-zero probability $p = \sin^2(\theta)$

$$A|0\rangle = |\Psi\rangle$$

$$|\Psi_1\rangle = \sum_{x \in X_1} \alpha_x |x\rangle$$

$$|\Psi_0\rangle = \sum_{y \in X_0} \alpha_y |y\rangle$$

Generalization: Amplitude Amplification
(BBHT,BH,BHT,G,BHMT,…)

Consider functions with t solutions

$$X_1 = f^{-1}(1) \qquad X_0 = f^{-1}(0) \qquad t = |X_1|$$

Consider any algorithm that works with non-zero probability $p = \sin^2(\theta)$

$$A|0\rangle = |\Psi\rangle$$

$$|\Psi\rangle = \sin(\theta)|\Psi_1\rangle + \cos(\theta)|\Psi_0\rangle$$

$$|\Psi_1\rangle = \sum_{x \in X_1} \alpha_x |x\rangle \qquad \sum_{x \in X_1} |\alpha_x|^2 = 1$$

$$|\Psi_0\rangle = \sum_{y \in X_0} \alpha_y |y\rangle \qquad \sum_{y \in X_0} |\alpha_y|^2 = 1$$

73

Generalization: Amplitude Amplification
(BBHT,BH,BHT,G,BHMT,…)

$$A|0\rangle = |\Psi\rangle \qquad |\Psi\rangle = \sin(\theta)|\Psi_1\rangle + \cos(\theta)|\Psi_0\rangle$$

$$Q = -AU_0A^{-1}U_f$$

Generalization: Amplitude Amplification
(BBHT,BH,BHT,G,BHMT,…)

$$A|0\rangle = |\Psi\rangle \qquad |\Psi\rangle = \sin(\theta)|\Psi_1\rangle + \cos(\theta)|\Psi_0\rangle$$

$$Q = -AU_0A^{-1}U_f$$

$$Q^k A|0\rangle = \sin((2k+1)\theta)|\psi_1\rangle + \cos((2k+1)\theta)|\psi_0\rangle$$

Generalization: Amplitude Amplification
(BBHT,BH,BHT,G,BHMT,…)

$$A|0\rangle = |\Psi\rangle \qquad |\Psi\rangle = \sin(\theta)|\Psi_1\rangle + \cos(\theta)|\Psi_0\rangle$$

$$Q = -AU_0A^{-1}U_f$$

$$Q^k A|0\rangle = \sin((2k+1)\theta)|\psi_1\rangle + \cos((2k+1)\theta)|\psi_0\rangle$$

Generalization: Amplitude Amplification
(BBHT,BH,BHT,G,BHMT,…)

$$A|0\rangle = |\Psi\rangle \qquad |\Psi\rangle = \sin(\theta)|\Psi_1\rangle + \cos(\theta)|\Psi_0\rangle$$

$$Q = -AU_0A^{-1}U_f$$

$$Q^k A|0\rangle = \sin((2k+1)\theta)|\psi_1\rangle + \cos((2k+1)\theta)|\psi_0\rangle$$

We need

$$k \approx \frac{\pi}{4\theta} - \frac{1}{2} \approx O\left(\frac{1}{\sqrt{p}}\right)$$

OVERVIEW OF OTHER QUANTUM ALGORITHMS

Amplitude estimation

- Given operators

$$A|0\rangle = |\Psi\rangle = \sin(\theta)|\Psi_1\rangle + \cos(\theta)|\Psi_0\rangle$$

$$U_f : |\Psi_1\rangle \mapsto -|\Psi_1\rangle$$

$$|\Psi_0\rangle \mapsto |\Psi_0\rangle$$

- Estimate

$$\sin^2(\theta)$$

OVERVIEW OF OTHER QUANTUM ALGORITHMS

Generalization of Simon's problem, order-finding and DLP: "Hidden subgroup problem"

- A unifying framework was developed for these problems

$$f : G \rightarrow X$$

Generalization of Simon's problem, order-finding and DLP: "Hidden subgroup problem"

- A unifying framework was developed for these problems

$$f : G \rightarrow X$$

$$f(x) = f(y) \quad \text{iff} \quad x + S = y + S$$

for some $S \leq G$

Generalization of Simon's problem, order-finding and DLP: "Hidden subgroup problem"

- A unifying framework was developed for these problems

$$f : G \rightarrow X$$

$$f(x) = f(y) \quad \text{iff} \quad x + S = y + S$$

for some $S \leq G$

- If G is Abelian, finitely generated, and represented in a reasonable way, we can efficiently find S.

Example (II)

Discrete Log of $b = a^k$ to base a :

$G = Z_r \times Z_r$ $\quad X$ any group

$f(x, y) = a^x b^y$

$K = \langle (k, -1) \rangle$

Example (III)

Self-shift equivalences:

$$G = GF(q)^n \qquad X = GF(q)[X_1, X_2, \ldots, X_n]$$

$$f(a_1, a_2, \ldots, a_n) = P(X_1 - a_1, \ldots, X_n - a_n)$$

$$K = \{(a_1, \ldots, a_n) : P(X_1 - a_1, \ldots, X_n - a_n) = P(X_1, \ldots, X_n)\}$$

Example (III)

Self-shift equivalences:

$$G = GF(q)^n \qquad X = GF(q)[X_1, X_2, \ldots, X_n]$$

$$f(a_1, a_2, \ldots, a_n) = P(X_1 - a_1, \ldots, X_n - a_n)$$

$$K = \{(a_1, \ldots, a_n) : P(X_1 - a_1, \ldots, X_n - a_n) = P(X_1, \ldots, X_n)\}$$

Other applications of Abelian HSP

Other applications of Abelian HSP

- Any finite Abelian group G is the direct sum of finite cyclic groups

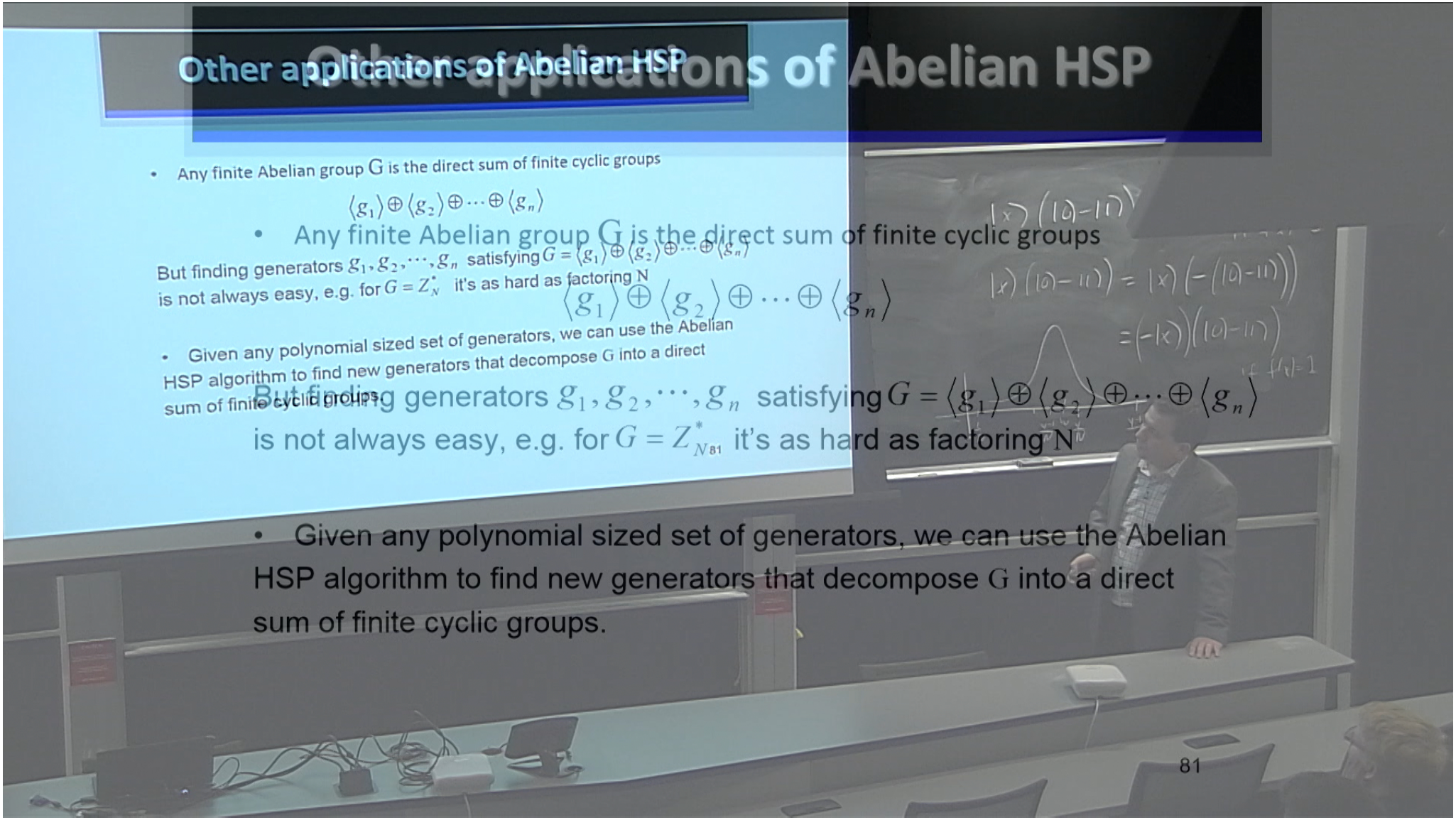
$$\langle g_1 \rangle \oplus \langle g_2 \rangle \oplus \cdots \oplus \langle g_n \rangle$$

Other applications of Abelian HSP

- Any finite Abelian group G is the direct sum of finite cyclic groups

$$\langle g_1 \rangle \oplus \langle g_2 \rangle \oplus \cdots \oplus \langle g_n \rangle$$

But finding generators $g_1, g_2, \cdots, g_n$ satisfying $G = \langle g_1 \rangle \oplus \langle g_2 \rangle \oplus \cdots \oplus \langle g_n \rangle$ is not always easy, e.g. for $G = Z_N^*$ it's as hard as factoring N


Other applications of Abelian HSP
Other applications of Abelian HSP
• Any finite Abelian group G is the direct sum of finite cyclic groups
$\langle g_1 \rangle \oplus \langle g_2 \rangle \oplus \cdots \oplus \langle g_n \rangle$
But finding generators $g_1, g_2, \cdots, g_n$ satisfying $G = \langle g_1 \rangle \oplus \langle g_2 \rangle \oplus \cdots \oplus \langle g_n \rangle$
is not always easy, e.g. for $G = Z_N^*$ it's as hard as factoring N
• Given any polynomial sized set of generators, we can use the Abelian HSP algorithm to find new generators that decompose G into a direct sum of finite cyclic groups.
• Any finite Abelian group G is the direct sum of finite cyclic groups
$\langle g_1 \rangle \oplus \langle g_2 \rangle \oplus \cdots \oplus \langle g_n \rangle$
But finding generators $g_1, g_2, \cdots, g_n$ satisfying $G = \langle g_1 \rangle \oplus \langle g_2 \rangle \oplus \cdots \oplus \langle g_n \rangle$
is not always easy, e.g. for $G = Z_{N81}^*$ it's as hard as factoring N
• Given any polynomial sized set of generators, we can use the Abelian HSP algorithm to find new generators that decompose G into a direct sum of finite cyclic groups.
81

What about non-Abelian HSP

- Consider the symmetric group $G = S_n$
- S_n is the set of permutations of n elements
- Let G be an n-vertex graph
- Let $X_G = \{\pi(G) \mid \pi \in S_n\}$
- Define $f_G : S_n \to X_G \qquad f_G(\pi) = \pi(G)$

What about non-Abelian HSP

- Consider the symmetric group $G = S_n$
- S_n is the set of permutations of n elements
- Let G be an n-vertex graph

- Let $X_G = \{\pi(G) \mid \pi \in S_n\}$
- Define $f_G : S_n \to X_G \qquad f_G(\pi) = \pi(G)$
- Then $f_G(\pi_1) = f_G(\pi_2) \Leftrightarrow \pi_1 K = \pi_2 K$

where $K = AUT(G) = \{\pi \mid \pi(G) = G\}$

Graph automorphism problem

- So the hidden subgroup of f_G is the automorphism group of G
- This is a difficult problem in NP that is believed not to be in BPP and yet not NP-complete.
- A solution to the graph automorphism problem gives a solution to the graph isomorphism problem.

Simulation of quantum mechanical systems

Input:

- $H = \sum_{k=1}^{L} H_k$

  where each H_k acts non-trivially on a subsystem of size bounded by a constant

Simulation of quantum mechanical systems

Input:

- $H = \sum_{k=1}^{L} H_k$

  where each H_k acts non-trivially on a subsystem of size bounded by a constant

- a description of an easy-to-prepare quantum state $|\Psi_0\rangle$

84

Simulation of quantum mechanical systems

Input:

- $H = \sum_{k=1}^{L} H_k$

  where each H_k acts non-trivially on a subsystem of size bounded by a constant

- a description of an easy-to-prepare quantum state $|\Psi_0\rangle$

- A positive accuracy δ

Simulation of quantum mechanical systems

Input:

- $H = \sum_{k=1}^{L} H_k$

  where each H_k acts non-trivially on a subsystem of size bounded by a constant

- a description of an easy-to-prepare quantum state $|\Psi_0\rangle$

- A positive accuracy δ

- A time t_f at which the evolved state is desired

Simulation Algorithm: Output and Runtime

- A state $\left|\widetilde{\Psi}(t_f)\right\rangle$ such that

$$\left|\left\langle \widetilde{\Psi}(t_f)\right| e^{-iHt_f}\left|\Psi_0\right\rangle\right|^2 \geq 1-\delta$$

Simulation Algorithm: Output and Runtime

- A state $\left|\widetilde{\Psi}(t_f)\right\rangle$ such that

$$\left|\left\langle \widetilde{\Psi}(t_f)\right| e^{-iHt_f}\left|\Psi_0\right\rangle\right|^2 \geq 1-\delta$$

- Runtime $O\left(poly\left(\frac{1}{\delta}, L\right)\right)$ steps