

Title: PSI 17/18 - Quantum Information - Lecture 6

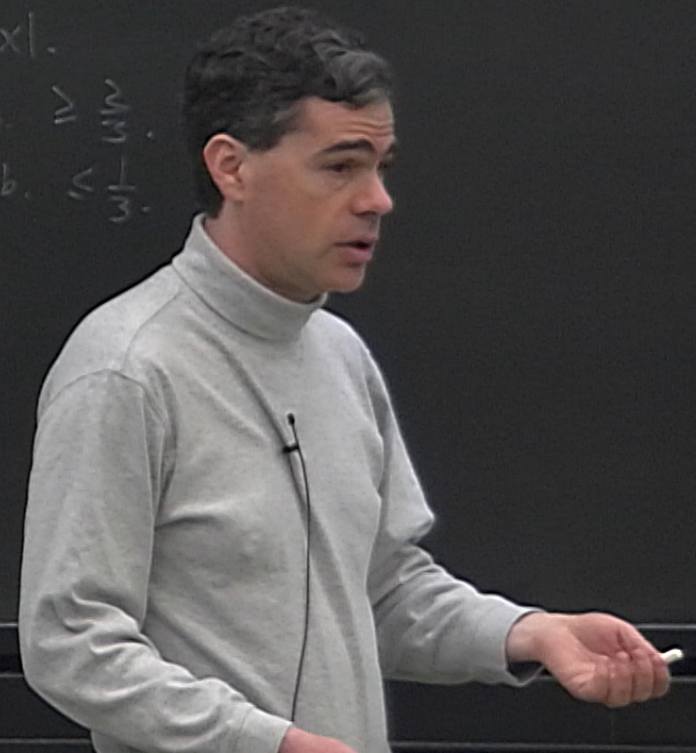
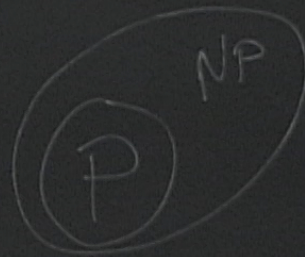
Date: Feb 27, 2018 11:30 AM

URL: <http://pirsa.org/18020049>

Abstract:

Def: BQP is the set of languages L
for which $\exists$ a quantum algorithm $A_L(x)$ st.

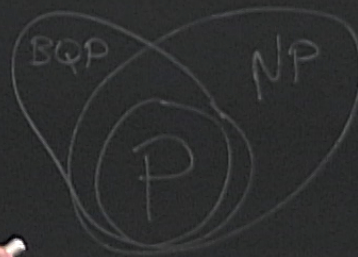
1. $A_L(x)$ runs in time polynomial in $|x|$.
2. If $x \in L$, then $A_L(x) = 1$ w/ prob. $\geq \frac{2}{3}$.
3. If $x \notin L$, then $A_L(x) = 1$ w/ prob. $\leq \frac{1}{3}$.



Def: BQP is the set of languages L
for which $\exists$ a quantum algorithm $A_L(x)$ st.
 $A_L(x)$ runs in time polynomial in $|x|$.

$\forall x \in L$, then $A_L(x) = 1$ w/ prob. $\geq \frac{2}{3}$.

$\forall x \notin L$, then $A_L(x) = 1$ w/ prob. $\leq \frac{1}{3}$.



Def.: In a promise problem, the input
(the instances) satisfy some constraint
that may be hard to check.

Def.: In a promise problem, the input (the instances) satisfy some constraint that may be hard to check.

Def.: A function $f(n)$ is $O(g(n))$ if there exists constants C, n_0 s.t.
 $f(n) \leq Cg(n)$ for $\forall n > n_0$.

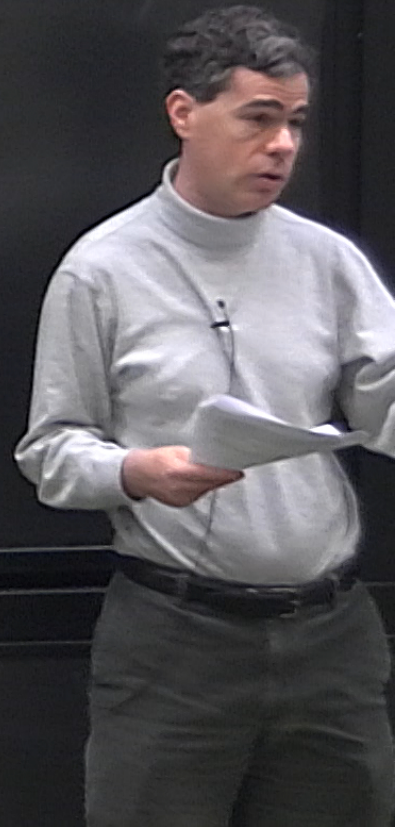
problem, the input
satisfy some constraint
to check.

$f(n)$ is $O(g(n))$
constants C, n_0 s.t.
 $\forall n > n_0$.

$$O: \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$$

Def.: Let $f(O)$ be a property of oracles.

The query complexity of f is the minimum
of queries to O that is needed to calculate $f(O)$.



Example: We are promised the $O: \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$
is either
constant $O(x) = O(y) \forall x, y$
or balanced $|\{x \mid O(x) = 0\}| = |\{y \mid O(y) = 1\}|$.

Determine which is true w/ 100%.

used the $0: \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$

$O(y) \forall x, y$

$$|O(x) = \vec{0}| = |\{y \mid O(y) = \vec{0}\}|.$$

e.g. 100%.

Classical algorithm:

Try $2^{n-1} + 1$ inputs

All the same $\Rightarrow$ constant

Any are different $\Rightarrow$ balanced.



problem, the input
satisfy some constraint
to check.

$f(n)$ is $O(g(n))$
constants c, n_0 s.t.
 $\forall n > n_0$.

$$O: \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$$

Def.: Let $f(O)$ be a property of oracles.

The query complexity of f is the minimum
of queries to O that is needed to calculate $f(O)$.

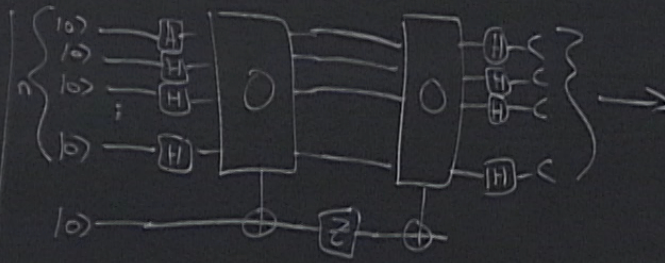
Quantum oracle
 $O: |x\rangle|a\rangle \mapsto |x\rangle|a \oplus O(x)\rangle$

classical algorithm:

Try $2^{n-1} + 1$ inputs

All the same $\Rightarrow$ constant

Any are different $\Rightarrow$ balanced.



All 0 $\Rightarrow$ "constant"
Anything else $\Rightarrow$ "balanced"

Example: We are promised the $O: \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$
is either
constant $O(x) = O(y) \quad \forall x, y$
or balanced $|\{x \mid O(x) = 0\}| = |\{y \mid O(y) = 1\}|$.

Determine which is true w/ 100%.

Classical algorithm:

Try $2^{n-1} + 1$ inputs

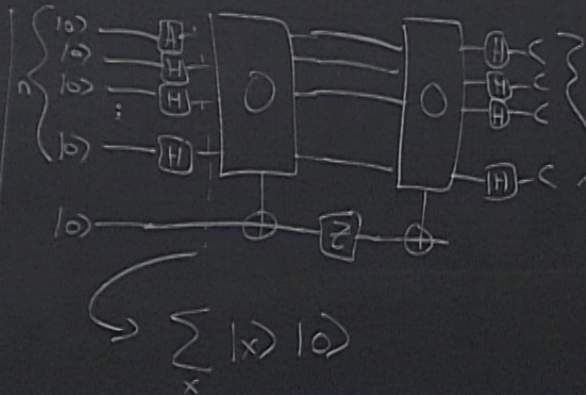
All the same $\Rightarrow$ constant

Any are different $\Rightarrow$ balanced.

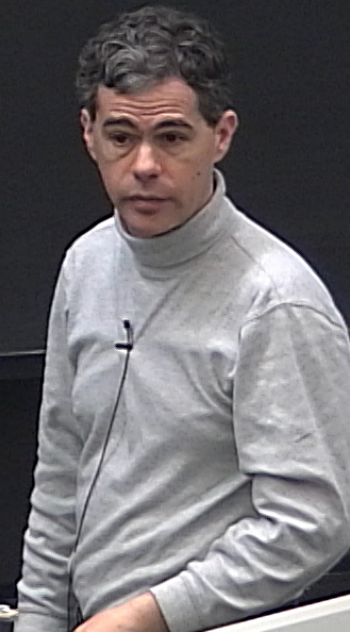
classical algorithm:

Try $2^{n-1} + 1$ inputs

All the same $\Rightarrow$ constant
 They are different $\Rightarrow$ balanced.



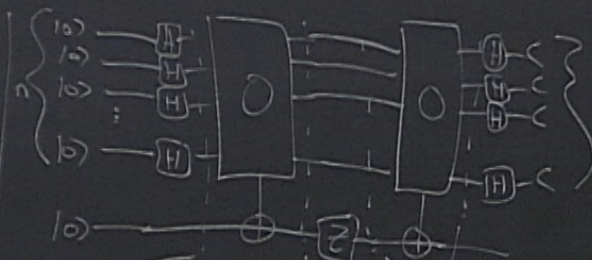
All 0 $\Rightarrow$ "constant"
 Anything else $\Rightarrow$ "balanced"



classical algorithm:

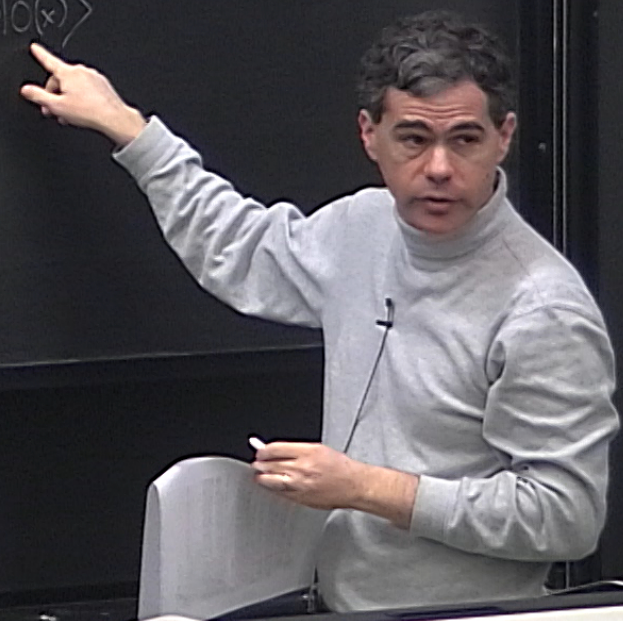
Try $2^{n-1} + 1$ inputs

All the same $\Rightarrow$ constant
 they are different $\Rightarrow$ balanced.



All 0 $\Rightarrow$ "constant"
 Anything else $\Rightarrow$ "balanced"

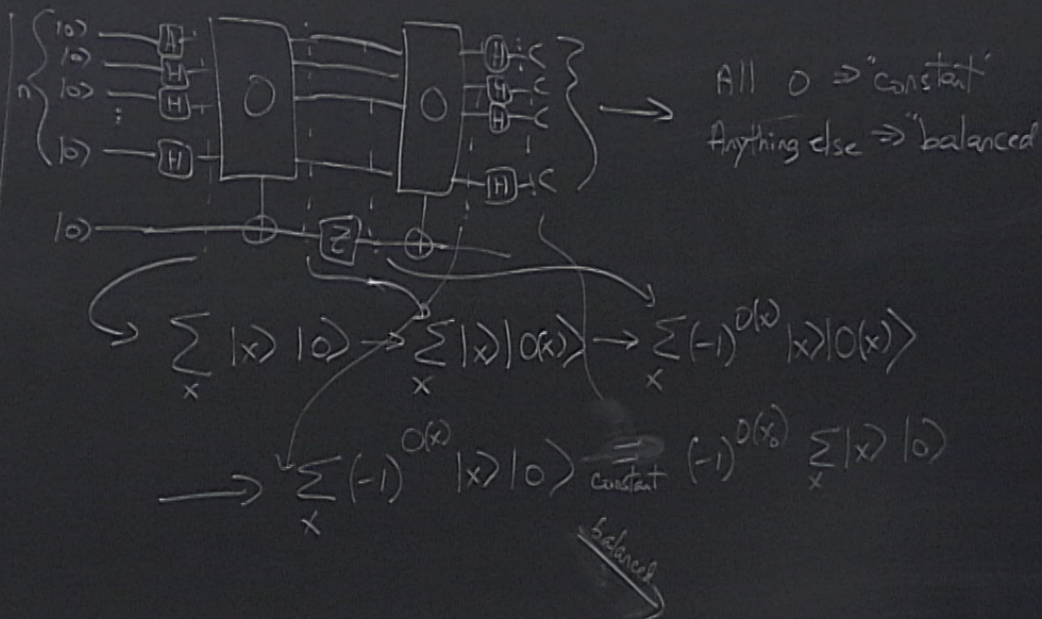
$$\sum_x |x\rangle |0\rangle \rightarrow \sum_x |x\rangle |f(x)\rangle \rightarrow \sum_x (-1)^{f(x)} |x\rangle |0(x)\rangle$$



classical algorithm:

Try $2^{n-1} + 1$ inputs

All the same $\Rightarrow$ constant
 They are different $\Rightarrow$ balanced.

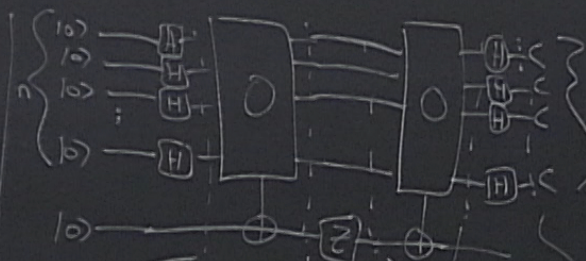


$$\begin{aligned} \sum_x |x\rangle |0\rangle &\rightarrow \sum_x |x\rangle |f(x)\rangle \rightarrow \sum_x (-1)^{f(x)} |x\rangle |0(x)\rangle \\ &\rightarrow \sum_x (-1)^{0(x)} |x\rangle |0\rangle \stackrel{\text{constant}}{=} (-1)^{0(x_0)} \sum_x |x\rangle |0\rangle \\ &\quad \swarrow \text{balanced} \end{aligned}$$

classical algorithm:

Try $2^{n-1} + 1$ inputs

All the same $\Rightarrow$ constant
 They are different $\Rightarrow$ balanced.



All 0 $\Rightarrow$ "constant"
 Anything else $\Rightarrow$ "balanced"

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

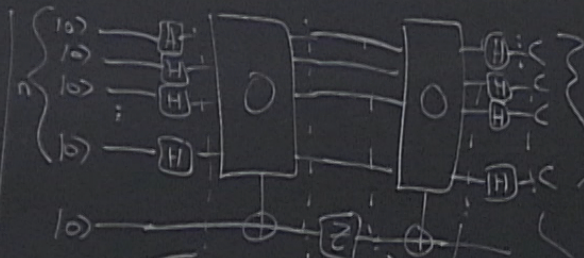
$$\rightarrow \sum_x (-1)^{f(x)} |x\rangle |0\rangle \xrightarrow{\text{constant}} (-1)^{f(x_0)} \sum_x |x\rangle |0\rangle \rightarrow (-1)^{f(x_0)} |0\rangle |0\rangle$$

$\xrightarrow{\text{balanced}}$

classical algorithm:

Try $2^{n-1} + 1$ inputs

All the same $\Rightarrow$ constant
 They are different $\Rightarrow$ balanced.



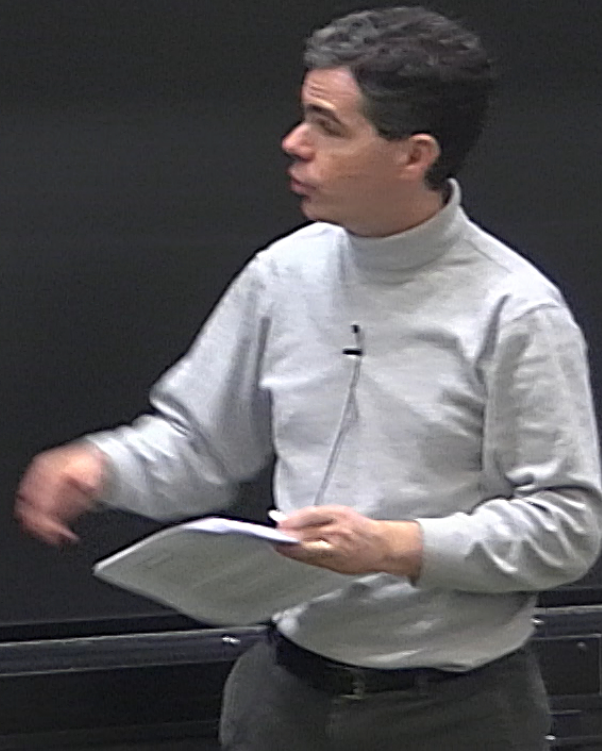
All 0 $\Rightarrow$ "constant"
 Anything else $\Rightarrow$ "balanced"

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

$$\rightarrow \sum_x (-1)^{f(x)} |x\rangle |0\rangle \xrightarrow{\text{constant}} (-1)^{f(0)} \sum_x |x\rangle |0\rangle \rightarrow (-1)^{f(0)} |0\rangle |0\rangle \checkmark$$

$\swarrow$
balanced
 $\searrow$

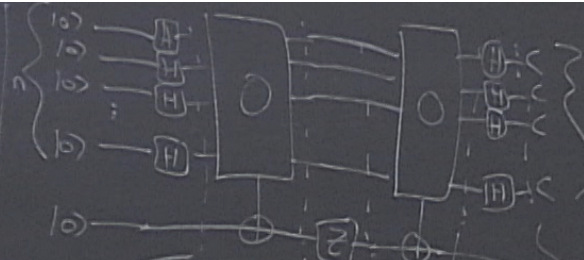
$$\begin{aligned}
 |4\rangle &= \sum_x |x\rangle \\
 |\phi\rangle &= \sum_x (-1)^{O(x)} |x\rangle \\
 \langle 4|\phi\rangle &= \sum_x (-1)^{O(x)} \langle x|x\rangle = 0
 \end{aligned}$$



classical algorithm:

Try $2^{n-1} + 1$ inputs

All the same $\Rightarrow$ constant
 Any are different $\Rightarrow$ balanced.



All 0 $\Rightarrow$ "constant"
 Anything else $\Rightarrow$ "balanced"

$$\sum_x |x\rangle |0\rangle \rightarrow \sum_x |x\rangle |O(x)\rangle \rightarrow \sum_x (-1)^{O(x)} |x\rangle |O(x)\rangle$$

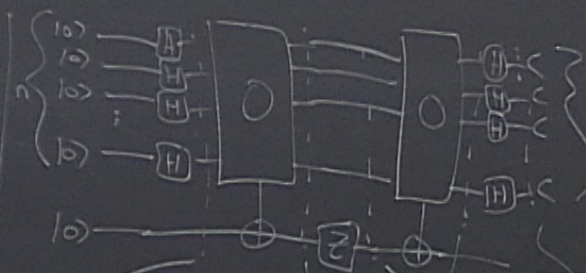
$$\rightarrow \sum_x (-1)^{O(x)} |x\rangle |0\rangle \xrightarrow{\text{constant}} (-1)^{O(x_0)} \sum_x |x\rangle |0\rangle \rightarrow (-1)^{O(x_0)} |0, 0\rangle |0\rangle \checkmark$$

$\xrightarrow{\text{balanced}} H^{\otimes n} |\phi\rangle |0\rangle$

classical algorithm:

Try $2^{n-1} + 1$ inputs

all the same $\Rightarrow$ constant
 they are different $\Rightarrow$ balanced.



All 0 $\Rightarrow$ "constant"
 Anything else $\Rightarrow$ "balanced"

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

$$\rightarrow \sum_x (-1)^{f(x)} |x\rangle |0\rangle \xrightarrow{\text{constant}} (-1)^{f(x_0)} \sum_x |x\rangle |0\rangle \rightarrow (-1)^{f(x_0)} |0\rangle |0\rangle \checkmark$$

balanced

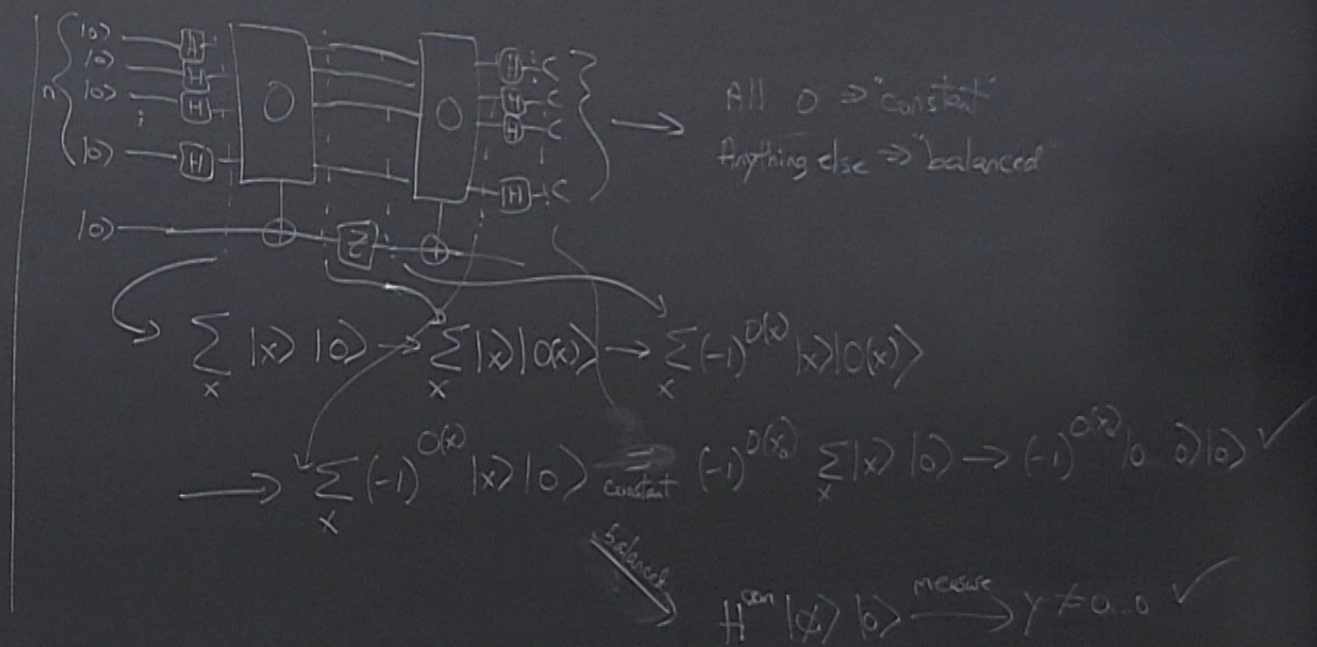
$$H^{\otimes n} |\phi\rangle |0\rangle \xrightarrow{\text{measure}} y \neq 0 \dots 0 \checkmark$$

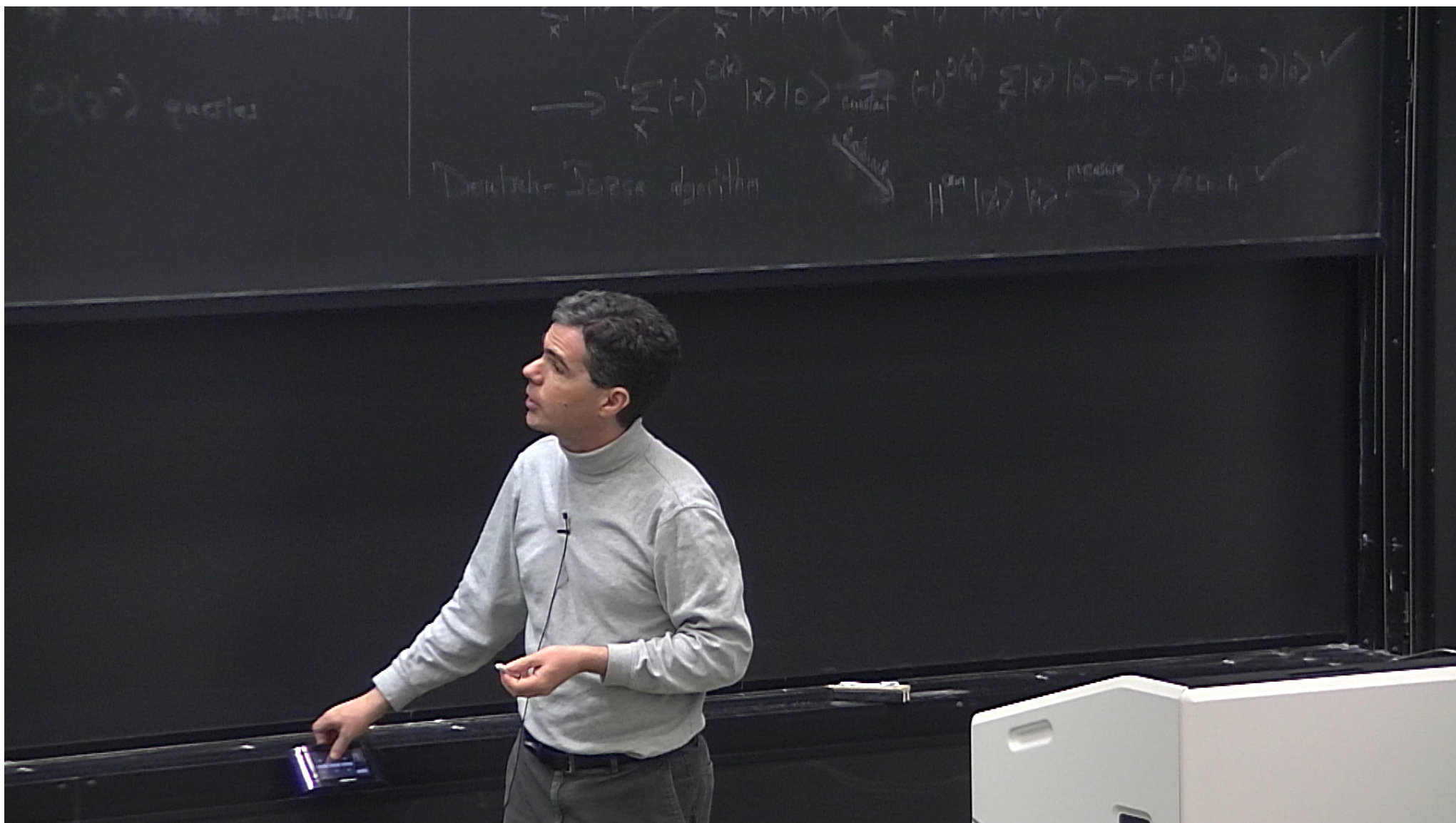
Classical algorithm:

Try $2^{n-1} + 1$ inputs

All the same $\Rightarrow$ constant
 They are different $\Rightarrow$ balanced.

$O(2^n)$ queries





$$|\psi\rangle = \sum_x |x\rangle$$

$$|\phi\rangle = \sum_x (-1)^{O(x)} |x\rangle$$

$$\langle \psi | \phi \rangle = \sum_x (-1)^{O(x)} \langle x | x \rangle = 0$$

$$\Rightarrow \underbrace{\langle \psi | H^{0,1}}_{\langle 0,0 |} \underbrace{(H^{0,1} | \phi \rangle)}_{\sum_{y \neq 0,0} \alpha_y | y \rangle} = 0$$

Factoring: Given N , find p & q such that $pq = N$.
Usually p & q prime.

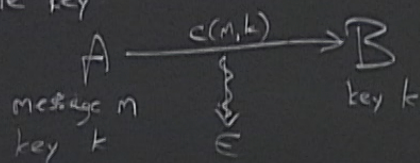
Factoring: Given N , find $p \& q$ such that $pq = N$.

Usually $p \& q$ prime.

- Decision version: Does N have a factor less than M ?
- In NP.
- Multiplication is $\text{poly}(\log N)$
- Best classical factoring alg. is $\exp((\log N)^{1/3})$

RSA public key cryptosystem

Private key:



$c(m, k)$ ciphertext

Bob decrypts $d(c, k)$ s.t.

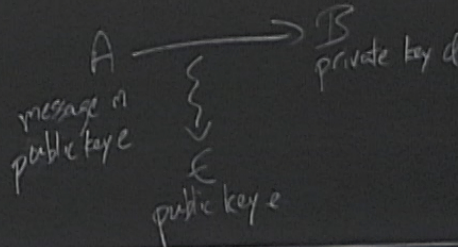
$$d(c(m, k), k) = m$$

Computational security

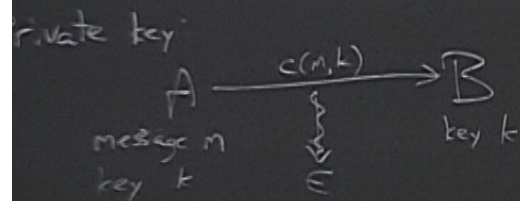
$$c(m, k) \rightarrow m$$

is exponentially hard
without knowledge of k .

Public key e , private d



SA public key cryptosystem



$c(m,k)$ ciphertext

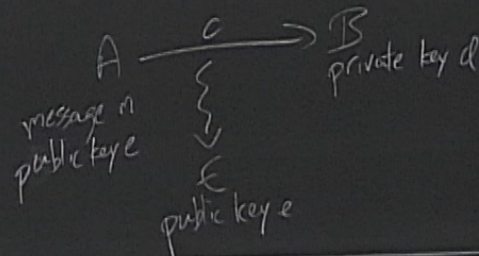
$\rightarrow$ decrypts $d(c,k)$ s.t.
 $d(c(m,k),k) = m$.

Computational security

$c(m,k) \rightarrow m$
 is exponentially hard
 without knowledge of k .

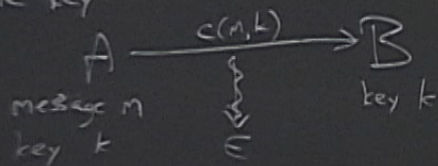
encrypt: $e, m \rightarrow$ ciphertext $c(e,m)$
 decrypt: $d, c \xrightarrow{D}$ decrypted text
 $D(d, c(e,m)) = m$

Public key e , private d



A public key cryptosystem

private key:



$c(m, k)$ ciphertext

decrypts $d(c, k)$ s.t.

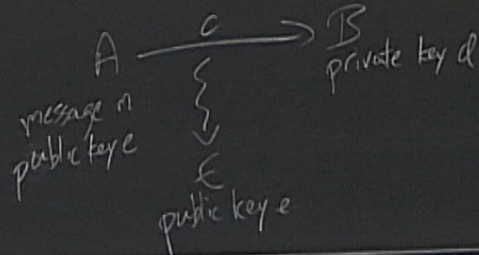
$$d(c(m, k), k) = m$$

Computational security

$$c(m, k) \rightarrow m$$

is exponentially hard
without knowledge of k .

Public key e , private d



encrypt: $e, m \rightarrow$ ciphertext $c(e, m)$

decrypt: $d, c \rightarrow$ decrypted text
 $D(d, c(e, m)) = m$

Computing $c(m, k), e \rightarrow m$
is computationally hard.