# Week 10: Quantum algorithms for search and counting

COMS 4281 (Fall 2025)

# Admin

1. Practice worksheet, quiz out.
2. Problem Set 2 out this weekend, due Nov 28.

# So far

So far, we have seen two examples of exponential quantum speedups:

- Simons Algorithm
- Order Finding/Factoring

These speedups are for **highly structured problems**.

# Unstructured search problem

**Given**: Oracle access to a black-box function $f : \{0,1\}^n \to \{0,1\}$

**Goal**: Output a **marked input** $x$ such that $f(x) = 1$, if one exists.

# Unstructured search problem

**Given**: Oracle access to a black-box function $f : \{0,1\}^n \to \{0,1\}$

**Goal**: Output a **marked input** $x$ such that $f(x) = 1$, if one exists.

Classical algorithms need $\Omega(2^n)$ queries to $f$ to find a marked input.

# Grover's algorithm

Grover's Algorithm can find a marked input with $O(\sqrt{2^n})$ queries to $f$.

Presented by Lov Grover in 1997 in paper called "*A fast quantum mechanical algorithm for database search*".

# Grover's algorithm

This achieves a **quadratic speedup**. Not as impressive as exponential speedup, but still interesting!

1 billion seconds $\approx$ 31 years

$\sqrt{1 \text{ billion}}$ seconds $\approx$ 9 hours.

# Phase versus XOR oracles

# Phase vs XOR oracles

The Simons and Deutsch algorithms called the oracle via the **XOR oracle**

$$U_f |x, b\rangle = |x, b \oplus f(x)\rangle \; .$$

# Phase vs XOR oracles

The Simons and Deutsch algorithms called the oracle via the **XOR oracle**

$$U_f |x, b\rangle = |x, b \oplus f(x)\rangle \ .$$

In Grover's algorithm the oracle is accessed via the **Phase oracle**

$$O_f |x\rangle = (-1)^{f(x)} |x\rangle \, .$$

Note that $O_f$ acts on $n$ **qubits**.

The Phase Oracle can be simulated by one query to the XOR oracle, and XOR oracle can be simulated with one query to the Phase Oracle.

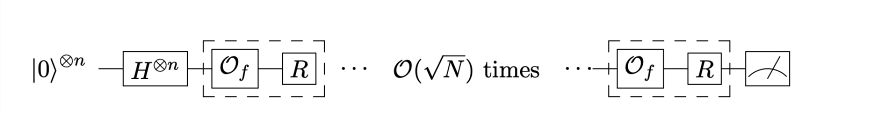
(do on board)

# Grover's algorithm

# Grover's algorithm

For simplicity, let's assume that there exists **exactly one** marked input.

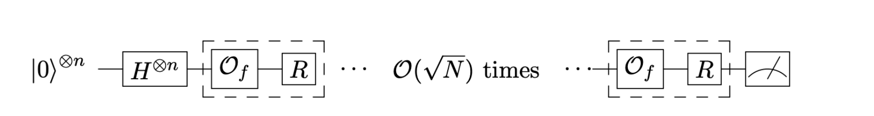
The circuit for Grover's algorithm:



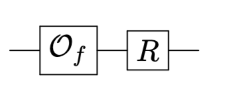
# Grover's algorithm

For simplicity, let's assume that there exists **exactly one** marked input.

The circuit for Grover's algorithm:




where




is the **Grover iterate**.

# Grover iterate

The state of Grover's algorithm in the very beginning is an equal superposition of all possible inputs:

$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \,.$$

The operator $R$ is the $n$-qubit **Grover diffusion operator**

$$R = 2\,|s\rangle\,\langle s| - \mathrm{I}$$

# The Grover diffusion operator

How to implement the Grover diffusion operator

$$R = 2\left|s\right\rangle\left\langle s\right| - \mathrm{I}$$

as a circuit composed of single and two-qubit gates?

## The Grover diffusion operator

How to implement the Grover diffusion operator

$$R = 2\,|s\rangle\,\langle s| - \mathrm{I}$$

as a circuit composed of single and two-qubit gates?

First, observe that

$$R = H^{\otimes n}(2\,|0^n\rangle\,\langle 0^n| - \mathrm{I})H^{\otimes n}\ .$$

# The Grover diffusion operator

How to implement the Grover diffusion operator

$$R = 2\left|s\right\rangle\left\langle s\right| - \mathrm{I}$$

as a circuit composed of single and two-qubit gates?

First, observe that

$$R = H^{\otimes n}(2\left|0^n\right\rangle\left\langle 0^n\right| - \mathrm{I})H^{\otimes n}\ .$$

Then, observe that

$$L = 2\left|0^n\right\rangle\left\langle 0^n\right| - \mathrm{I}$$

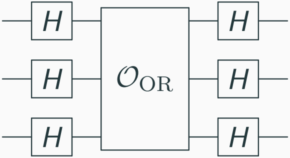
is the *phase oracle* $\mathcal{O}_{\mathrm{OR}}$ corresponding to the boolean function $\mathrm{OR}(y_1, \ldots, y_n)$ that equals 1 when at least one of the $y_j$'s is 1.

# The Grover diffusion operator

We can implement the Grover diffusion operator

$$R = 2\,|s\rangle\,\langle s| - \mathrm{I}$$

as a circuit composed of single and two-qubit gates as follows:




(Here just shown for $n = 3$ qubits).

# Analysis of Grover's algorithm

Let $x^*$ denote the unique marked input.

**Important fact**: The intermediate states of Grover's algorithm are linear combinations of

$$|x^*\rangle \qquad \text{and} \qquad |\Delta\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq x^*} |x\rangle$$

where $N = 2^n$.

We can prove this via induction.

# Analysis of Grover's algorithm

Let $N = 2^n$.

**Base case**: initial state

$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_x |x\rangle = \sqrt{\frac{N-1}{N}}\, |\Delta\rangle + \frac{1}{\sqrt{N}}\, |x^*\rangle$$

# Analysis of Grover's algorithm

Let $N = 2^n$.

**Base case**: initial state

$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_x |x\rangle = \sqrt{\frac{N-1}{N}} |\Delta\rangle + \frac{1}{\sqrt{N}} |x^*\rangle$$

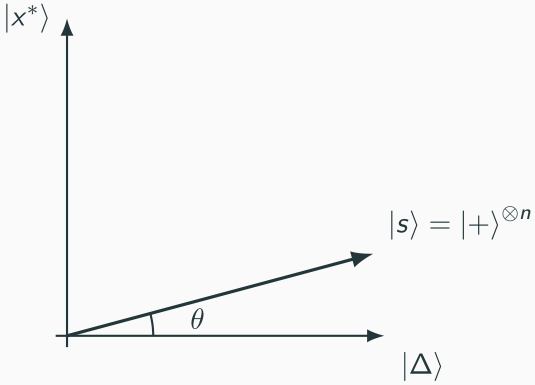
Let $\theta = \sin^{-1}\left(\frac{1}{\sqrt{N}}\right)$. Then

$$|s\rangle = \cos(\theta) |\Delta\rangle + \sin(\theta) |x^*\rangle \ .$$

# Analysis of Grover's algorithm

We can visualize the state of the algorithm in the 2D subspace spanned by $|\Delta\rangle$ and $|x^*\rangle$ as follows.

**Initial state**:



## Analysis of Grover's algorithm

Suppose $|\psi\rangle = \cos(\tau)\,|\Delta\rangle + \sin(\tau)\,|x^*\rangle$ is the intermediate state of the Grover algorithm after some number of iterations.

The next Grover iterate first applies the phase oracle $O_f$, and then applies $R = 2\,|s\rangle\,\langle s| - \mathrm{I}$.

## Analysis of Grover's algorithm

Suppose $|\psi\rangle = \cos(\tau)\,|\Delta\rangle + \sin(\tau)\,|x^*\rangle$ is the intermediate state of the Grover algorithm after some number of iterations.

The next Grover iterate first applies the phase oracle $O_f$, and then applies $R = 2\,|s\rangle\,\langle s| - \mathrm{I}$.

**Claim**: Geometrically, $O_f$ **reflects** the vector $|\psi\rangle$ across the horizontal $|\Delta\rangle$ axis.

# Analysis of Grover's algorithm

Suppose $|\psi\rangle = \cos(\tau)|\Delta\rangle + \sin(\tau)|x^*\rangle$ is the intermediate state of the Grover algorithm after some number of iterations.

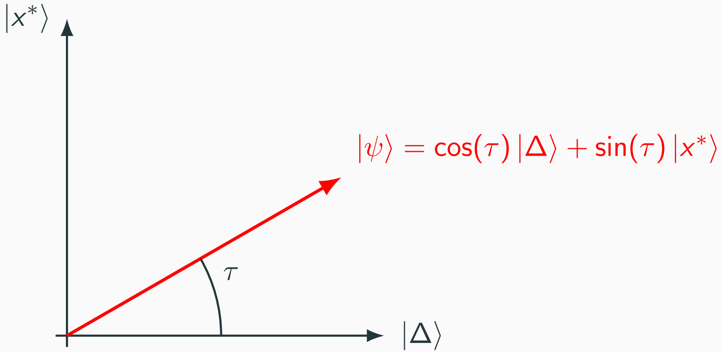
The next Grover iterate first applies the phase oracle $O_f$, and then applies $R = 2|s\rangle\langle s| - \mathrm{I}$.

**Claim**: Geometrically, $O_f$ **reflects** the vector $|\psi\rangle$ across the horizontal $|\Delta\rangle$ axis.

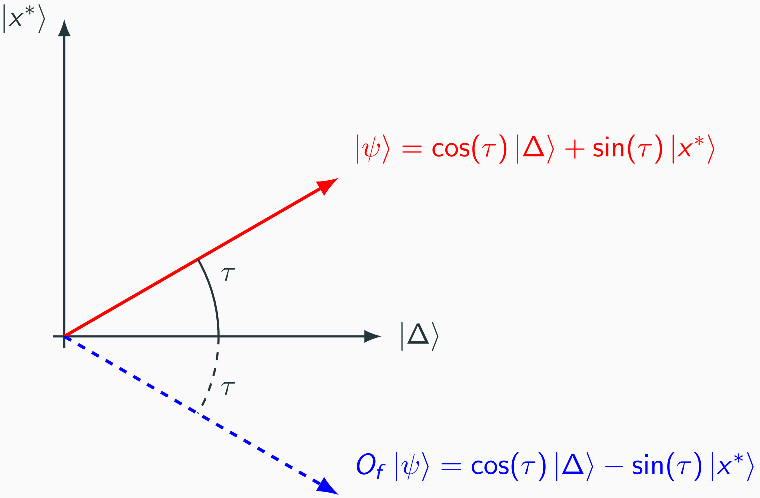
**Proof**:

$$O_f|\psi\rangle = \cos(\tau)|\Delta\rangle - \sin(\tau)|x^*\rangle.$$

# Analysis of Grover's algorithm



# Analysis of Grover's algorithm



# Analysis of Grover's algorithm

**Claim**: Geometrically, $R = 2\left|s\right\rangle\left\langle s\right| - \mathrm{I}$ **reflects** vectors across $\left|s\right\rangle$ axis.

## Analysis of Grover's algorithm

**Claim**: Geometrically, $R = 2\,|s\rangle\,\langle s| - \mathrm{I}$ **reflects** vectors across $|s\rangle$ axis.

**Proof**: Let $|\varphi\rangle = \alpha\,|s\rangle + \beta\,|s^{\perp}\rangle$ be an arbitrary vector. Then

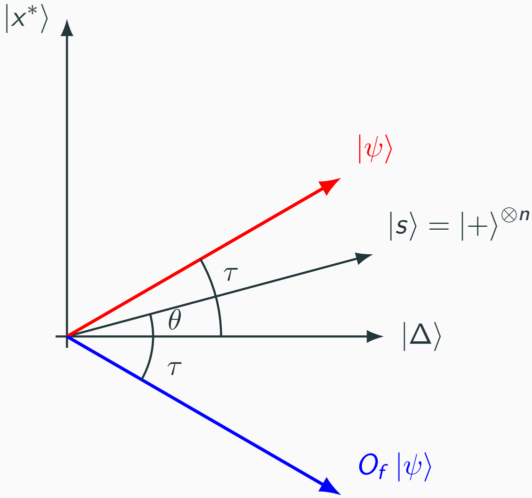
$$\begin{aligned} R\,|\varphi\rangle &= (2\,|s\rangle\,\langle s| - \mathrm{I})(\alpha\,|s\rangle + \beta\,|s^{\perp}\rangle) \\ &= \alpha(2\,|s\rangle\,\langle s \mid s\rangle - |s\rangle) + \beta(2\,|s\rangle\,\langle s \mid s^{\perp}\rangle - |s^{\perp}\rangle) \end{aligned}$$

**Claim**: Geometrically, $R = 2\left|s\right\rangle\left\langle s\right| - \mathrm{I}$ **reflects** vectors across $\left|s\right\rangle$ axis.

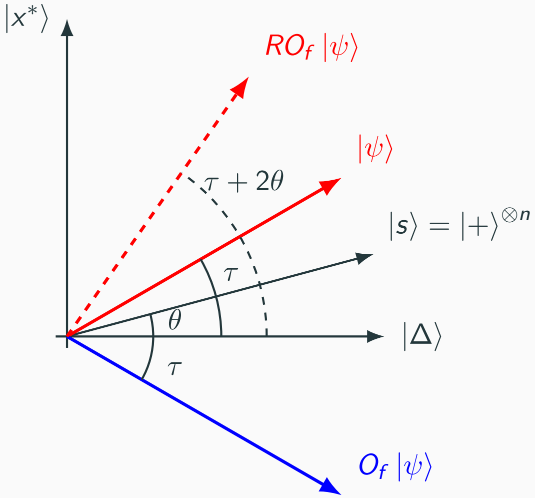
**Proof**: Let $\left|\varphi\right\rangle = \alpha\left|s\right\rangle + \beta\left|s^{\perp}\right\rangle$ be an arbitrary vector. Then

$$
\begin{aligned}
R\left|\varphi\right\rangle &= (2\left|s\right\rangle\left\langle s\right| - \mathrm{I})(\alpha\left|s\right\rangle + \beta\left|s^{\perp}\right\rangle) \\
&= \alpha(2\left|s\right\rangle\left\langle s \mid s\right\rangle - \left|s\right\rangle) + \beta(2\left|s\right\rangle\left\langle s \mid s^{\perp}\right\rangle - \left|s^{\perp}\right\rangle) \\
&= \alpha\left|s\right\rangle - \beta\left|s^{\perp}\right\rangle
\end{aligned}
$$

# Analysis of Grover's algorithm



# Analysis of Grover's algorithm



# Analysis of Grover's algorithm

**Claim**: After $k$ Grover iterations, the state of the algorithm is

$$|\psi_k\rangle = \cos((2k+1)\theta)\,|\Delta\rangle + \sin((2k+1)\theta)\,|x^*\rangle$$

where $\theta = \sin^{-1}(\sqrt{1/N})$.

# Analysis of Grover's algorithm

**Claim**: After $k$ Grover iterations, the state of the algorithm is

$$|\psi_k\rangle = \cos((2k+1)\theta)\,|\Delta\rangle + \sin((2k+1)\theta)\,|x^*\rangle$$

where $\theta = \sin^{-1}(\sqrt{1/N})$.

How many iterations to run so that measuring $|\psi_k\rangle$ yields the solution $|x^*\rangle$ with high probability?

**Claim**: After $k$ Grover iterations, the state of the algorithm is

$$|\psi_k\rangle = \cos((2k+1)\theta)\,|\Delta\rangle + \sin((2k+1)\theta)\,|x^*\rangle$$

where $\theta = \sin^{-1}(\sqrt{1/N})$.

How many iterations to run so that measuring $|\psi_k\rangle$ yields the solution $|x^*\rangle$ with high probability?

We want to pick $k$ so that

$$\sin((2k+1)\theta) \approx 1\ .$$

This means $(2k+1)\theta \approx \pi/2$.

# Analysis of Grover's algorithm

**Claim**: After $k$ Grover iterations, the state of the algorithm is

$$|\psi_k\rangle = \cos((2k+1)\theta)\,|\Delta\rangle + \sin((2k+1)\theta)\,|x^*\rangle$$

where $\theta = \sin^{-1}(\sqrt{1/N})$.

How many iterations to run so that measuring $|\psi_k\rangle$ yields the solution $|x^*\rangle$ with high probability?

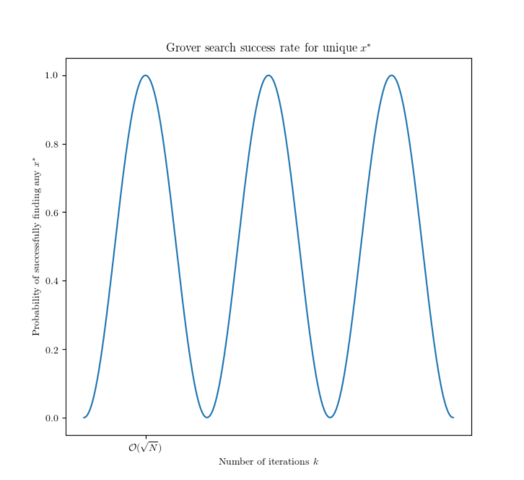
We want to pick $k$ so that

$$\sin((2k+1)\theta) \approx 1 \ .$$

This means $(2k+1)\theta \approx \pi/2$. Using the small angle approximation, we have that $\theta = \sin^{-1}(1/\sqrt{N}) \approx 1/\sqrt{N}$, so $k \approx \pi\sqrt{N}/4$.

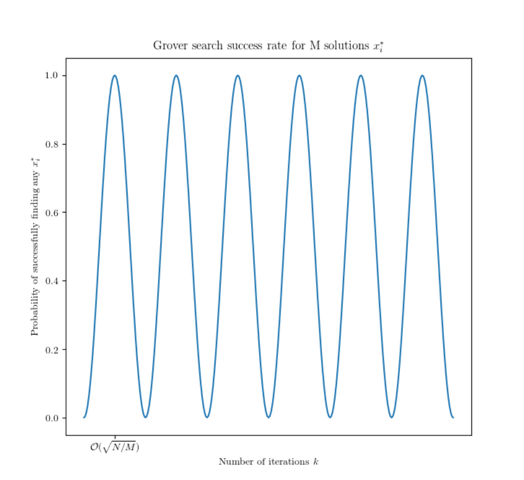
# Success probability curve

The probability of finding a marked input when measuring the state after $k$ Grover iterations. Running Grover's algorithm for longer can reduce your success probability!



# Multiple solutions

What happens when there are $M > 1$ solutions to $f(x) = 1$? We can use Grover's algorithm as before, but the success probability curve changes:



## Multiple solutions

If there are $M > 1$ solutions, then can find a solution with $O(\sqrt{N/M})$ queries.

# Multiple solutions

If there are $M > 1$ solutions, then can find a solution with $O(\sqrt{N/M})$ queries.

The intermediate states of the algorithm are in the span of

- $|\Gamma\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} |x\rangle$, uniform superposition over all **solutions**
- $|\Delta\rangle = \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} |x\rangle$, uniform superposition over all **non-solutions**

In the end, the output is a **uniformly random** solution.

# Multiple solutions

What if you wanted to output **all** solutions?

There is $O(\sqrt{NM})$ query solution:

1. Use $\sqrt{\frac{N}{M}}$ queries to find the first solution $x_1$.

## Multiple solutions

What if you wanted to output **all** solutions?

There is $O(\sqrt{NM})$ query solution:

1. Use $\sqrt{\frac{N}{M}}$ queries to find the first solution $x_1$.
2. Run Grover search with updated oracle $f_1$ where $x_1$ is excluded. This finds solution $x_2$ with $\sqrt{\frac{N}{M-1}}$ queries.

# Multiple solutions

What if you wanted to output **all** solutions?

There is $O(\sqrt{NM})$ query solution:

1. Use $\sqrt{\frac{N}{M}}$ queries to find the first solution $x_1$.
2. Run Grover search with updated oracle $f_1$ where $x_1$ is excluded. This finds solution $x_2$ with $\sqrt{\frac{N}{M-1}}$ queries.
3. Update the oracle to exclude $x_2$. Find another solution $x_3$, etc.

# Multiple solutions

The total number of queries is

$$\sqrt{\frac{N}{M}} + \sqrt{\frac{N}{M-1}} + \cdots + \sqrt{\frac{N}{1}}$$

# Multiple solutions

The total number of queries is

$$\sqrt{\frac{N}{M}} + \sqrt{\frac{N}{M-1}} + \cdots + \sqrt{\frac{N}{1}}$$

$$= \sum_{j=0}^{M-1} \sqrt{\frac{N}{M-j}} \leq \int_0^{M-1} \sqrt{\frac{N}{M-x}} \mathrm{d}x$$

$$\leq O(\sqrt{NM})$$

# Use cases for Grover search

Contrary to Grover's original title, "database search" is probably not a good application of Grover's algorithm.

# Use cases for Grover search

Contrary to Grover's original title, "database search" is probably not a good application of Grover's algorithm.

For database search, the oracle $f(x)$ would correspond to something like "If Person $x$ lives in New York City, has blood type O and likes Thai food, output 1" where $1 \leq x \leq N$.

Although Grover's algorithm makes $O(\sqrt{N})$ queries to $f$, computing $f(x)$ itself may have complexity $N = 2^n$.

# Use cases for Grover search

A much better use is when we're trying to solve an abstract search problem where $f(x)$ can be computed much less than $N = 2^n$ time.

**Example**: Inverting hash functions (such as SHA-3).

A hash function $H : \{0, 1\}^n \to \{0, 1\}^k$ maps long strings to shorter strings (a "hash"). Used everywhere in cryptography and computer security.

# Use cases for Grover search

A much better use is when we're trying to solve an abstract search problem where $f(x)$ can be computed much less than $N = 2^n$ time.

**Example**: Inverting hash functions (such as SHA-3).

A hash function $H : \{0,1\}^n \to \{0,1\}^k$ maps long strings to shorter strings (a "hash"). Used everywhere in cryptography and computer security.

It is believed that certain hash functions, such as SHA-3, are hard to **invert**: that is, given an **image** $y \in \{0,1\}^k$, finding an $x \in \{0,1\}^n$ such that $H(x) = y$ should require $\sim \exp(n)$ time.

## Use cases for Grover search

Mining bitcoin and other cryptocurrencies all boil down to spending enormous compute power to invert hash functions.

# Use cases for Grover search

Mining bitcoin and other cryptocurrencies all boil down to spending enormous compute power to invert hash functions.

With Grover, can invert hash functions in $O(\sqrt{2^n}) \cdot \mathrm{poly}(n)$ time on a quantum computer.

## Use cases for Grover search

Mining bitcoin and other cryptocurrencies all boil down to spending enormous compute power to invert hash functions.

With Grover, can invert hash functions in $O(\sqrt{2^n}) \cdot \mathrm{poly}(n)$ time on a quantum computer.

Given a hash function $H : \{0,1\}^n \to \{0,1\}^k$ that can be computed in the forwards direction in $\mathrm{poly}(n)$ time (such as SHA-3), an image $y \in \{0,1\}^k$, a quantum computer can apply Grover to the following oracle $f : \{0,1\}^n \to \{0,1\}$:

$$f(x) = 1 \text{ if and only if } H(x) = y \ .$$

## Use cases for Grover search

Mining bitcoin and other cryptocurrencies all boil down to spending enormous compute power to invert hash functions.

With Grover, can invert hash functions in $O(\sqrt{2^n}) \cdot \mathrm{poly}(n)$ time on a quantum computer.

Given a hash function $H : \{0,1\}^n \to \{0,1\}^k$ that can be computed in the forwards direction in $\mathrm{poly}(n)$ time (such as SHA-3), an image $y \in \{0,1\}^k$, a quantum computer can apply Grover to the following oracle $f : \{0,1\}^n \to \{0,1\}$:

$$f(x) = 1 \text{ if and only if } H(x) = y \ .$$

Grover will call the oracle for $f$ a total of $O(\sqrt{2^n})$ times. Each oracle call has complexity $\mathrm{poly}(n)$.

# Use cases for Grover search

Thus, quantum computers can give a **quadratic** speedup for many generic search problems.

This does not have as dramatic/drastic effect on cryptography like Shor's algorithm does, but it still has repercussions for security (e.g., key lengths have to be doubled to recover the original levels of security).

# Next time

Quantum Counting.