# CPEN 400Q Lecture 20
# The oracle, query complexity, and Deutsch's algorithm

Friday 24 March 2023

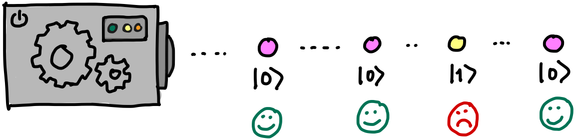
Apr 7/10: no class !!

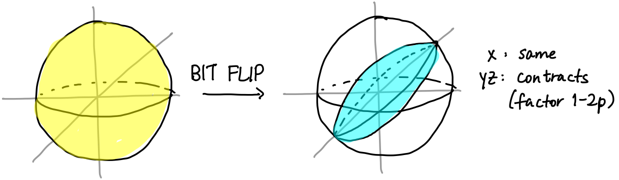
- Project schedule posted on Piazza
- Full grading rubric posted in PrairieLearn
- Quiz 9 (last quiz) on Monday about *today's* material
- Literacy assignment 3 due 29 March (Wednesday) at 23:59
- Assignment 3 available Monday (2 questions)

# Last time

We formalized the idea of quantum channels and discussed some common error channels

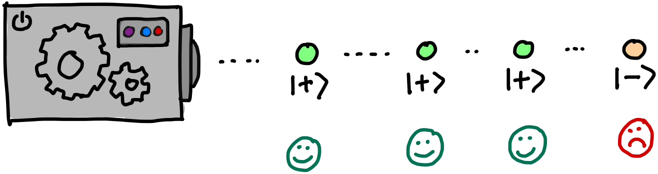
$$\mathcal{E}(\rho) = (1-p)\cdot\rho + p\cdot X\rho X$$

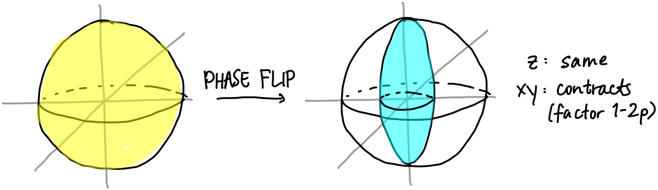


# Last time

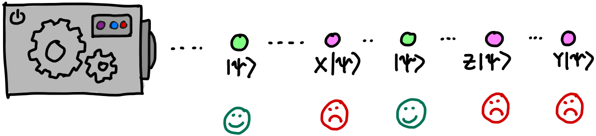
$$\mathcal{E}(\rho) = (1-p)\cdot\rho + p\cdot Z\rho Z$$

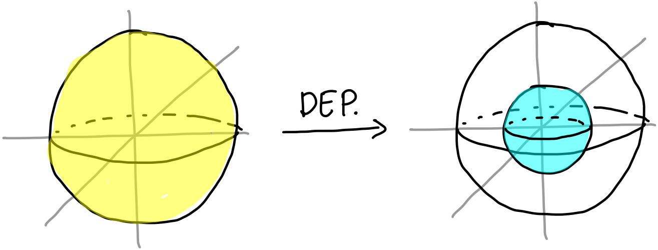


# The depolarizing channel

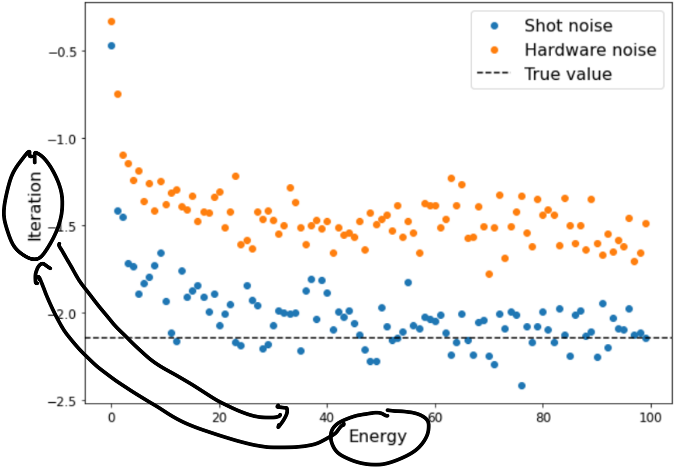
$$\mathcal{E}(\rho) = (1-p)\cdot\rho + \frac{p}{3}\cdot X\rho X + \frac{p}{3} Y\rho Y + \frac{p}{3} Z\rho Z$$



# The depolarizing channel

We discussed metrics for quality of noisy hardware, and did a more realistic run of the deuteron VQE using simulated hardware noise.

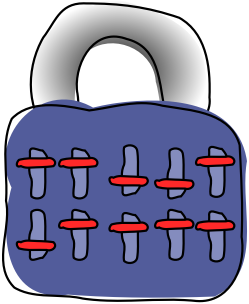



To make this better, see the video lecture on Canvas about *error mitigation*.

# Learning outcomes

- Define the query complexity of an algorithm
- Describe multiple strategies for incorporating an *oracle* query into a quantum circuit
- Implement Deutsch's algorithm in PennyLane

Suppose we would like to find the combination for a “binary” lock:



How do we solve this classically?

Try all combinations

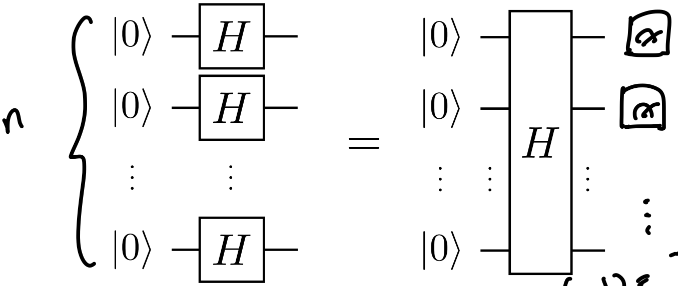
$\downarrow$

$O(2^n)$

Image credit: Codebook node A.1

Can we do better with a quantum computer?

What if we take $n$ qubits and put them in a superposition with all possible combinations?

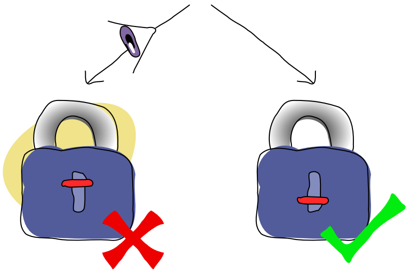



$\text{Prob (sol)} = \frac{1}{2^n}$

Often called the *Hadamard transform*.

# Idea: use superposition

Measurements are probabilistic - just because we put things into a uniform superposition of states, and our solution is "in" there, doesn't mean we are any closer to solving our problem.



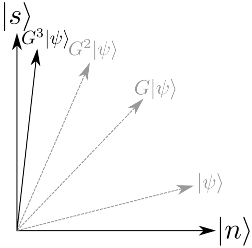
Quantum computers are **NOT** faster because they can "compute everything at the same time."

Image credit: Codebook node A.1

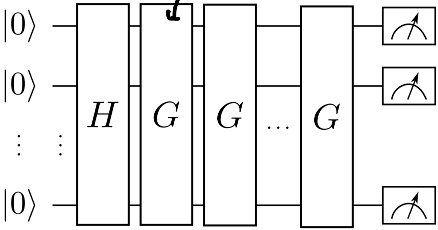
# Solving problems with quantum computers

*Can* we solve this problem better with a quantum computer?

Yes: **amplitude amplification**, and **Grover's algorithm**

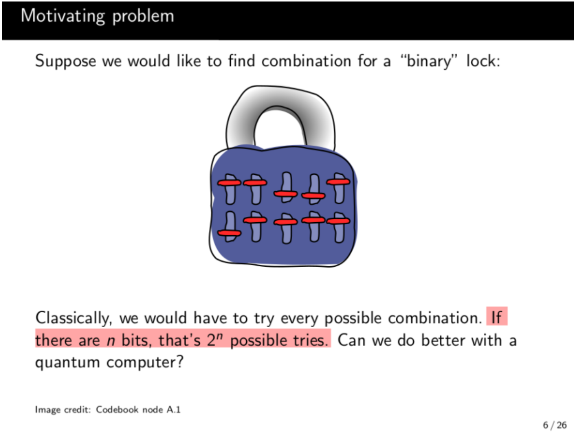



grover "iteration"




We will explore the algorithmic primitives that are involved, and some other cases where we can do better with quantum computing.

# Oracles




What is a "try"?

# Oracles

We often express these tries as **evaluations** of a function that tells us whether we have found the correct answer.

Let

- **x** be an *n*-bit string that represents an input to the lock
- **s** be the solution to the problem (i.e., the correct combination)

We can represent trying a lock combination as a function:

$$f(\vec{x}) = \begin{cases} 1 & \vec{x} = \vec{s} \\ 0 & \text{otherwise} \end{cases}$$

# Oracles

We don't necessarily care *how* this function gets evaluated, only that it gives us an answer (more specifically, a yes/no answer).

$$f(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} = \mathbf{s} \\ 0 & \text{otherwise.} \end{cases}$$

We consider this function as a black box, or an **oracle**.

Every time we try a lock combination, we are **querying the oracle**. The amount of queries we make is the **query complexity**.

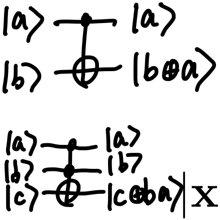
# Quantum oracles

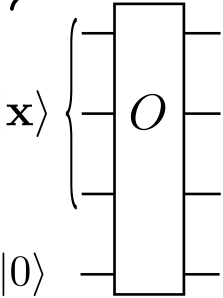
To solve this problem using quantum computing, we need some circuit that plays the role of the oracle.

Idea 1: encode the result in the state of an additional qubit.

$$O(|\vec{x}\rangle|y\rangle) = |\vec{x}\rangle|y \oplus f(\vec{x})\rangle$$

addition mod 2



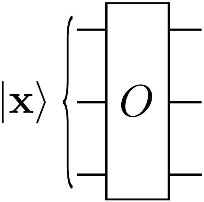




$$O|000\rangle|0\rangle = |000\rangle|0\rangle$$
$$O|001\rangle|0\rangle = |001\rangle|0\rangle$$
$$O|010\rangle|0\rangle = |010\rangle|0\rangle$$
$$O|011\rangle|0\rangle = |011\rangle|0\rangle$$
$$O|100\rangle|0\rangle = |100\rangle|0\rangle$$
$$O|101\rangle|0\rangle = |101\rangle|0\rangle$$
$$O|110\rangle|0\rangle = |110\rangle|1\rangle$$
$$O|111\rangle|0\rangle = |111\rangle|0\rangle$$

# Quantum oracles

Idea 2: encode the result in the phase of a qubit.

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




$$
\begin{aligned}
O|000\rangle &= |000\rangle \\
O|001\rangle &= |001\rangle \\
O|010\rangle &= |010\rangle \\
O|011\rangle &= |011\rangle \\
O|100\rangle &= |100\rangle \\
O|101\rangle &= |101\rangle \\
O|110\rangle &= -|110\rangle \\
O|111\rangle &= |111\rangle
\end{aligned}
$$

**Motivation**: You are given access to an oracle and are promised that it implements one of the following 4 functions:

| Name | Action | Name | Action | |
|---|---|---|---|---|
| $f_1$ | $f_1(0) = 0$<br>$f_1(1) = 0$ | $f_2$ | $f_2(0) = 1$<br>$f_2(1) = 1$ | constant |
| $f_3$ | $f_3(0) = 0$<br>$f_3(1) = 1$ | $f_4$ | $f_4(0) = 1$<br>$f_4(1) = 0$ | balanced |

Functions $f_1$ and $f_2$ are *constant* (same output no matter what the input), and $f_3$ and $f_4$ are balanced.

How many queries do you need to make to the oracle to determine if the function is constant or balanced? (i.e., either one of $f_1/f_2$, or one of $f_3/f_4$).

| Name | Action | Name | Action |
|---|---|---|---|
| $f_1$ | $f_1(0) = 0$<br>$f_1(1) = 0$ | $f_2$ | $f_2(0) = 1$<br>$f_2(1) = 1$ |
| $f_3$ | $f_3(0) = 0$<br>$f_3(1) = 1$ | $f_4$ | $f_4(0) = 1$<br>$f_4(1) = 0$ |

How many queries do you need to make to the oracle to determine if the function is constant or balanced? (i.e., either one of $f_1/f_2$, or one of $f_3/f_4$).

| Name | Action | Name | Action |
|---|---|---|---|
| $f_1$ | $f_1(0) = 0$<br>$f_1(1) = 0$ | $f_2$ | $f_2(0) = 1$<br>$f_2(1) = 1$ |
| $f_3$ | $f_3(0) = 0$<br>$f_3(1) = 1$ | $f_4$ | $f_4(0) = 1$<br>$f_4(1) = 0$ |

**Classical solution:** 2
We always need to query both inputs 0 and 1 to find out the nature of the function.

# Deutsch's algorithm

How many queries do you need to make to the oracle to determine if the function is constant or balanced? (i.e., either one of $f_1/f_2$, or one of $f_3/f_4$).

| Name | Action | Name | Action |
|---|---|---|---|
| $f_1$ | $f_1(0)=0$<br>$f_1(1)=0$ | $f_2$ | $f_2(0)=1$<br>$f_2(1)=1$ |
| $f_3$ | $f_3(0)=0$<br>$f_3(1)=1$ | $f_4$ | $f_4(0)=1$<br>$f_4(1)=0$ |

| | |
|---|---|
| 00 | 0 |
| 01 | 0 |
| 10 | 1 |
| 11 | 1 |

$$\frac{2^n}{2}+1$$

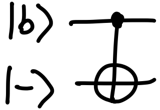
n-bit classical.

**Quantum solution:** 1
How???

The secret relies on *phase kickback*.

Remember: what happens when we apply a CNOT to the following two two-qubit states?

$$|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right),\quad |1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$




We get

$$\begin{aligned}
CNOT\left(|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right) &= |0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\\
CNOT\left(|1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right) &= |1\rangle\underbrace{\left(\frac{|1\rangle-|0\rangle}{\sqrt{2}}\right)}\\
&= |1\rangle(-|-\rangle)\\
&= (-|1\rangle)\,|-\rangle
\end{aligned}$$

$$O\left(|\vec{x}\rangle\right) = (-1)^{f(\vec{x})}|\vec{x}\rangle$$

↳ oracle

We can write a general version of this effect:

$$CNOT\left(|b\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\right) = (-1)^b|b\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

But what does this have to do with Deutsch's algorithm and figuring out if a function is constant or balanced?

This is where our oracle comes in. Suppose we have a black box, $U_f$, that implements any of these four functions, $f$:

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

put in superposition

Setting $|y\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ will allow us to 'extract' the value of $f(0) \oplus f(1)$ with just a single query.

$f(0) \oplus f(1) = 0$ : const.

$f(0) \oplus f(1) = 1$ : balanced

Let's work through the math.

$$U_f|\vec{x}\rangle|y\rangle = |\vec{x}\rangle|y \oplus f(\vec{x})\rangle$$

$$U_f|x\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2}}\left(U_f|\vec{x}\rangle|0\rangle - U_f|\vec{x}\rangle|1\rangle\right)$$

$$= \frac{1}{\sqrt{2}}\left(|\vec{x}\rangle|0 \oplus f(\vec{x})\rangle - |\vec{x}\rangle|1 \oplus f(\vec{x})\rangle\right)$$

If $f(x) = 0$, we get

$$U_f|x\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = (-1)^{f(\vec{x})}|\vec{x}\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

If $f(x) = 1$, we get

$$U_f|x\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = (-1)^{f(\vec{x})}|\vec{x}\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

$\boxed{U_f}$

So just like the case of the CNOT where we wrote the general version

$$CNOT\left(|b\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right)=(-1)^b|b\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right),$$

we can write

$$U_f\left(|\vec{x}\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right)=(-1)^{f(\vec{x})}|\vec{x}\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

Essentially, before the CNOT was just playing the role of $U_f$ for the specific function $f(0)=0, f(1)=1$.

This doesn't look like much on its own - we want to get a *combination* of $f(0)$ and $f(1)$. How can we do this?

By setting $|x\rangle$ to be a superposition!

$$U_f\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

$$= U_f\frac{|0\rangle}{\sqrt{2}}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)+U_f\frac{|1\rangle}{\sqrt{2}}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

$$= \underbrace{(-1)^{f(0)}}\left[\frac{|0\rangle}{\sqrt{2}}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)+(-1)^{f(1)}\frac{|1\rangle}{\sqrt{2}}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right]$$

$f(0)$

Let's pull out a phase factor of $(-1)^{f(0)}$, since global phase doesn't matter anyways.

$\underline{f(0) \oplus f(1)}$

# Deutsch's algorithm

$$= (-1)^{f(0)}\left[\frac{|0\rangle}{\sqrt{2}}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) + (-1)^{f(0)\oplus f(1)}\frac{|1\rangle}{\sqrt{2}}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right]$$

$$= \left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

Now let's look at how this state is different when $f$ is a constant vs. a balanced function.

$$|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$$

$$U_f\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)=\frac{|0\rangle+(-1)^{f(0)\oplus f(1)}|1\rangle}{\sqrt{2}}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

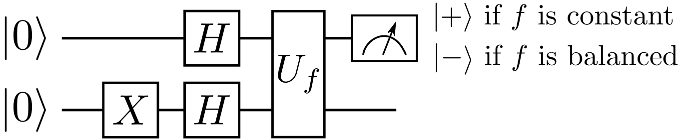
If the function is constant, $f(0) \oplus f(1) = 0$ and the state is

$$U_f\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)=\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

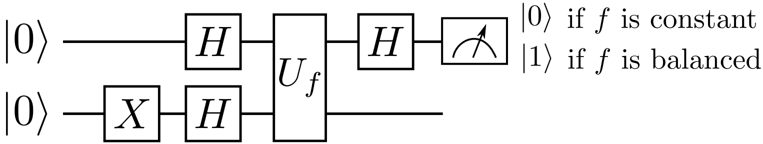
But if the function is balanced, $f(0) \oplus f(1) = 1$ and the state is

$$U_f\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)=\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

As a circuit, Deutsch's algorithm looks like this:

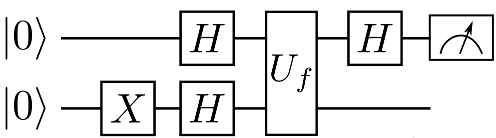



Or equivalently,

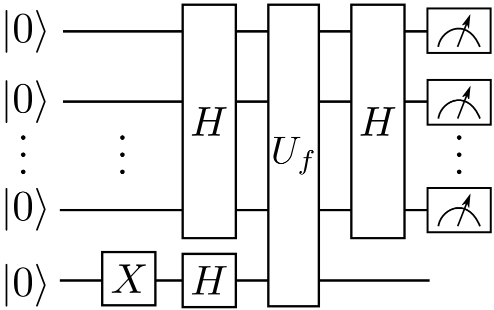



We call $U_f$ just once, but obtain information about the relationship between $f(0)$ and $f(1)$! Let's implement it.

# Generalization: Deutsch-Jozsa algorithm



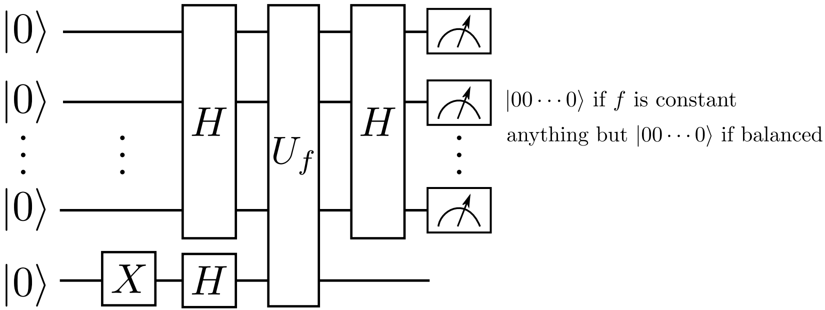

$|0\rangle$ if $f$ is constant
$|1\rangle$ if $f$ is balanced




$|00\cdots0\rangle$ if $f$ is constant
anything but $|00\cdots0\rangle$ if balanced

# Generalization: Deutsch-Jozsa algorithm

$2^{n-1} + 1$ classical queries in worst case; still only 1 quantum query.



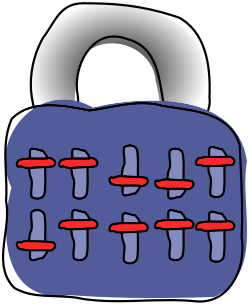

(Challenge: try implementing it yourself to check if this works!)

A few other interesting ones to look into in addition to Deutsch-Josza:

**Bernstein-Vazirani algorithm** (will see on A3)
Given $f : \{0,1\}^n \to \{0,1\}$ such that $f(x) = x \cdot s$ for some secret bitstring $s$. Find $s$ using the fewest number of queries to the oracle.

**Simon's algorithm**
Given $f : \{0,1\}^n \to \{0,1\}^n$ and promised that for some non-trivial bit string $s$, $f(x) = f(y)$ iff $x \oplus y = s$. Find $s$ using the fewest queries to the oracle

Let's break that lock!

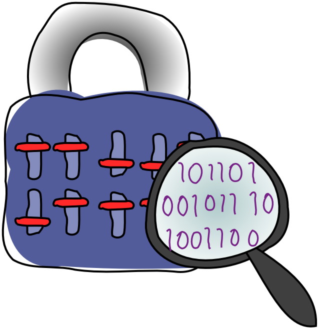


We input the combination to the lock as an $n$-bit (binary) string.
The correct combination is labelled $\mathbf{s}$.

Image credit: Codebook node A.1

# Grover's quantum search algorithm

How many times must we query the oracle to find the solution?




Classical: in the worst case, $2^n$ times

Quantum: $O(\sqrt{2^n})$ times

Idea: start with a uniform superposition and then *amplify* the amplitude of the state corresponding to the solution.

In other words, go from the uniform superposition

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{\vec{x} \in \{0,1\}^n} |\vec{x}\rangle$$

to something that looks more like this:

$$|\psi'\rangle = (\text{big number})\,|\vec{s}\rangle + (\text{small number}) \sum_{\vec{x} \neq \vec{s}} |\vec{x}\rangle$$

Q: Why do we want a state of this form?

$$|\psi'\rangle = (\text{big number})|\mathbf{s}\rangle + (\text{small number}) \sum_{\mathbf{x} \neq \mathbf{s}} |\mathbf{x}\rangle$$

Q: Why do we want a state of this form?

$$|\psi'\rangle = (\text{big number})|\mathbf{s}\rangle + (\text{small number}) \sum_{\mathbf{x} \neq \mathbf{s}} |\mathbf{x}\rangle$$

A: When we make a measurement, we will very likely get the solution to our problem!

We will see how to do this next time!

# Next time

Last few classes:

- Amplitude amplification, Grover's algorithm

Action items:

1. Literacy assignment 3
2. Project code and presentation

Recommended reading:

- Codebook modules A and G