CPEN 400Q / EECE 571Q Lecture 07
Oracles and Grover's algorithm

Tuesday 1 February 2022

Announcements

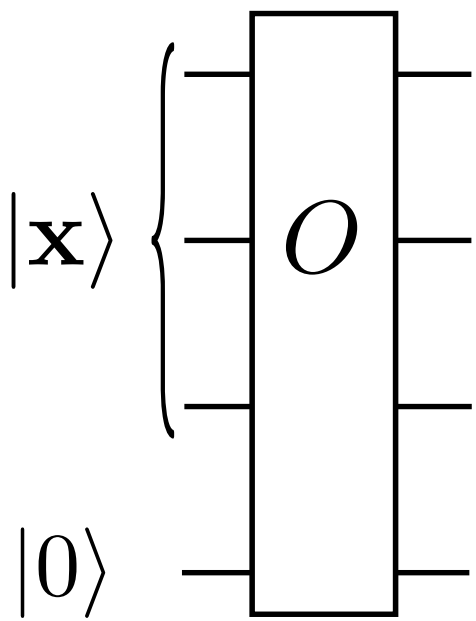
- Assignment 1 grades posted (see me or TA for questions)
- Assignment 2 now available (due Monday 14 Feb 23:59)
- Project details coming next week
- New Xanadu Quantum Codebook modules available today

Quiz 3 after class today.

Last time

We learned about oracles and query complexity

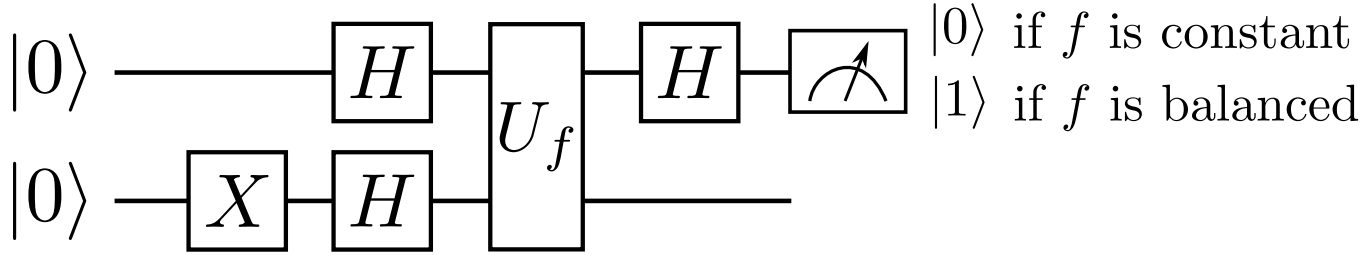
$$O|\mathbf{x}\rangle|y\rangle = |\mathbf{x}\rangle|y \oplus f(\mathbf{x})\rangle$$



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

Last time

We learned about phase kickback, and how it can be applied in Deutsch's algorithm to solve a small problem on a quantum computer using fewer oracle queries.



Learning outcomes

- Implement basic oracle circuits in PennyLane
- Describe the strategy of amplitude amplification
- Visualize Grover's algorithm in two different ways
- Perform quantum search with Grover's algorithm

Deutsch's algorithm

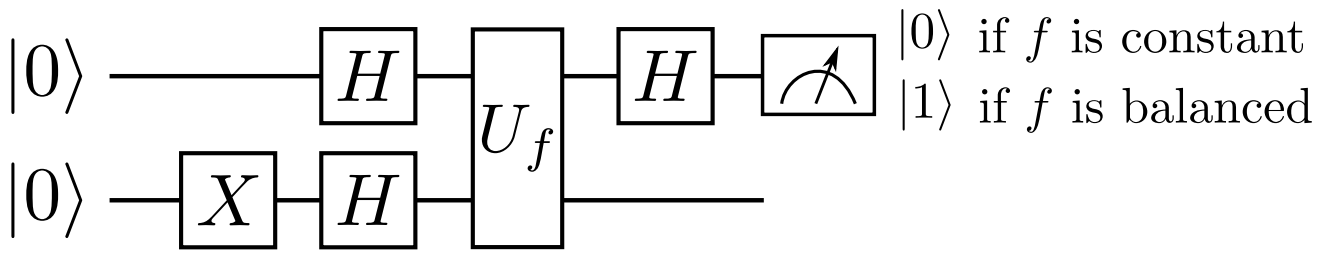
Recall that in Deutsch's algorithm, we are given an oracle and are promised that it implements one of the following 4 functions:

Name	Action	Name	Action
f_1	$f_1(0) = 0$ $f_1(1) = 0$	f_2	$f_2(0) = 1$ $f_2(1) = 1$
f_3	$f_3(0) = 0$ $f_3(1) = 1$	f_4	$f_4(0) = 1$ $f_4(1) = 0$

Functions f_1 and f_2 are *constant*, and f_3 and f_4 are balanced. We want to figure out what type of function we have.

Deutsch's algorithm

We solved this problem using the following circuit:



And we implemented it in PennyLane:

```
def deutsch_circuit(func=None):
    qml.PauliX(wires=1)
    qml.Hadamard(wires=0)
    qml.Hadamard(wires=1)

    oracle(func=func)

    qml.Hadamard(wires=0)

    return qml.probs(wires=0)
```

How do we implement the `oracle` function?

Deutsch's algorithm

The oracle has the following action in general for a function f:

$$U|\mathbf{x}\rangle|y\rangle = |\mathbf{x}\rangle|y \oplus f(\mathbf{x})\rangle$$

Case 1 ($f = f_1$)

$$f_1(0) = 0$$
$$f_1(1) = 0$$

How does the oracle act on the computational basis states?

$$U_{f_1}|0\rangle|0\rangle = |0\rangle|0 \oplus f_1(0)\rangle = |0\rangle|0 \oplus 0\rangle = |0\rangle|0\rangle$$
$$U_{f_1}|0\rangle|1\rangle = |0\rangle|1 \oplus f_1(0)\rangle = |0\rangle|1 \oplus 0\rangle = |0\rangle|1\rangle$$
$$U_{f_1}|1\rangle|0\rangle = |1\rangle|0 \oplus f_1(1)\rangle = |1\rangle|0 \oplus 0\rangle = |1\rangle|0\rangle$$
$$U_{f_1}|1\rangle|1\rangle = |1\rangle|1 \oplus f_1(1)\rangle = |1\rangle|1 \oplus 0\rangle = |1\rangle|1\rangle$$

Since the outcome is 0 in both cases, there is no effect on the basis states.

Case 2 ($f = f_2$)

$$f_2(0) = 1$$
$$f_2(1) = 1$$

Check computational basis states...

$$U_{f_2}|0\rangle|0\rangle = |0\rangle|0 \oplus f_2(0)\rangle =$$
$$U_{f_2}|0\rangle|1\rangle = |0\rangle|1 \oplus f_2(0)\rangle =$$
$$U_{f_2}|1\rangle|0\rangle = |1\rangle|0 \oplus f_2(1)\rangle =$$
$$U_{f_2}|1\rangle|1\rangle = |1\rangle|1 \oplus f_2(1)\rangle =$$

Case 3 ($f = f_3$)

$$f_3(0) = 0$$
$$f_3(1) = 1$$

Check computational basis states...

$$U_{f_3}|0\rangle|0\rangle = |0\rangle|0 \oplus f_3(0)\rangle =$$
$$U_{f_3}|0\rangle|1\rangle = |0\rangle|1 \oplus f_3(0)\rangle =$$
$$U_{f_3}|1\rangle|0\rangle = |1\rangle|0 \oplus f_3(1)\rangle =$$
$$U_{f_3}|1\rangle|1\rangle = |1\rangle|1 \oplus f_3(1)\rangle =$$

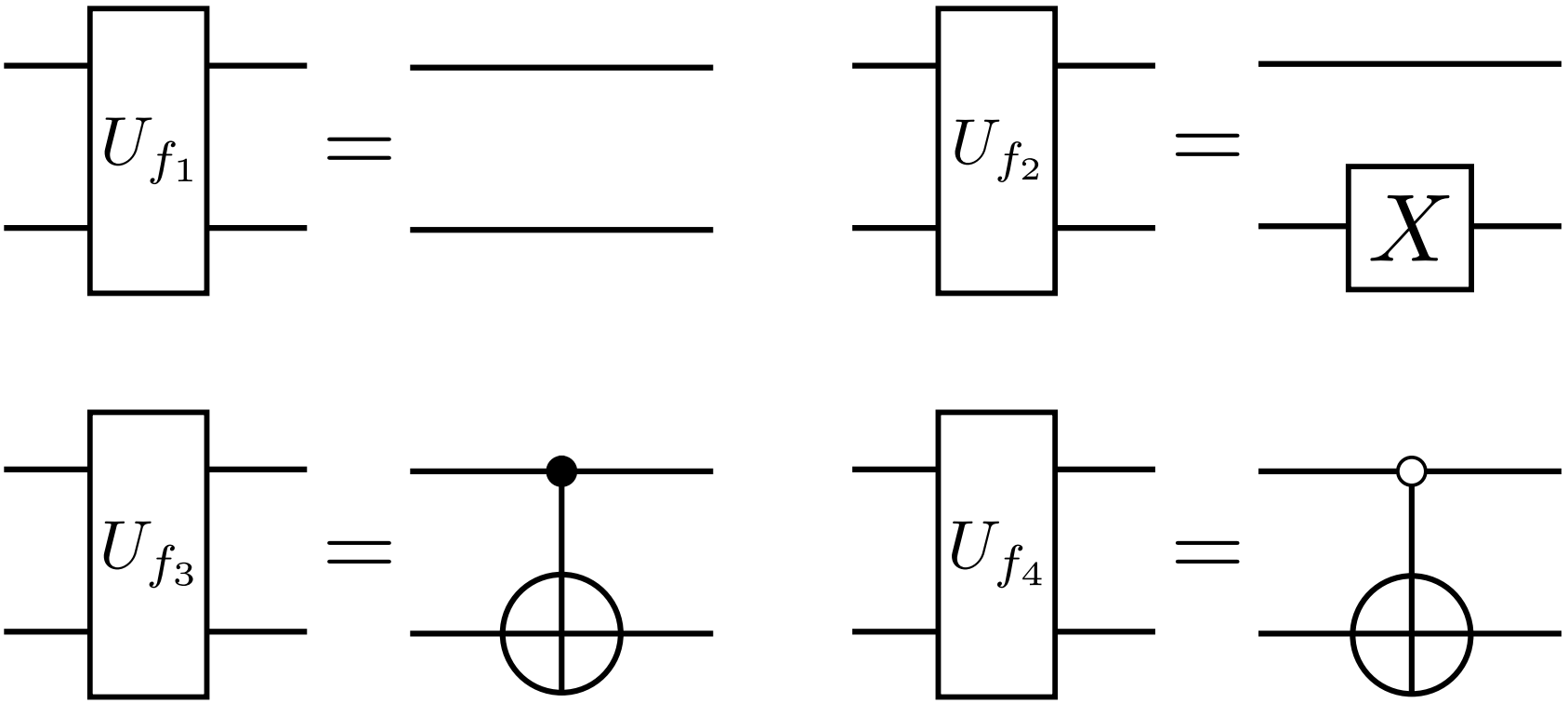
Case 3 ($f = f_4$)

$$f_4(0) = 1$$
$$f_4(1) = 0$$

Check computational basis states...

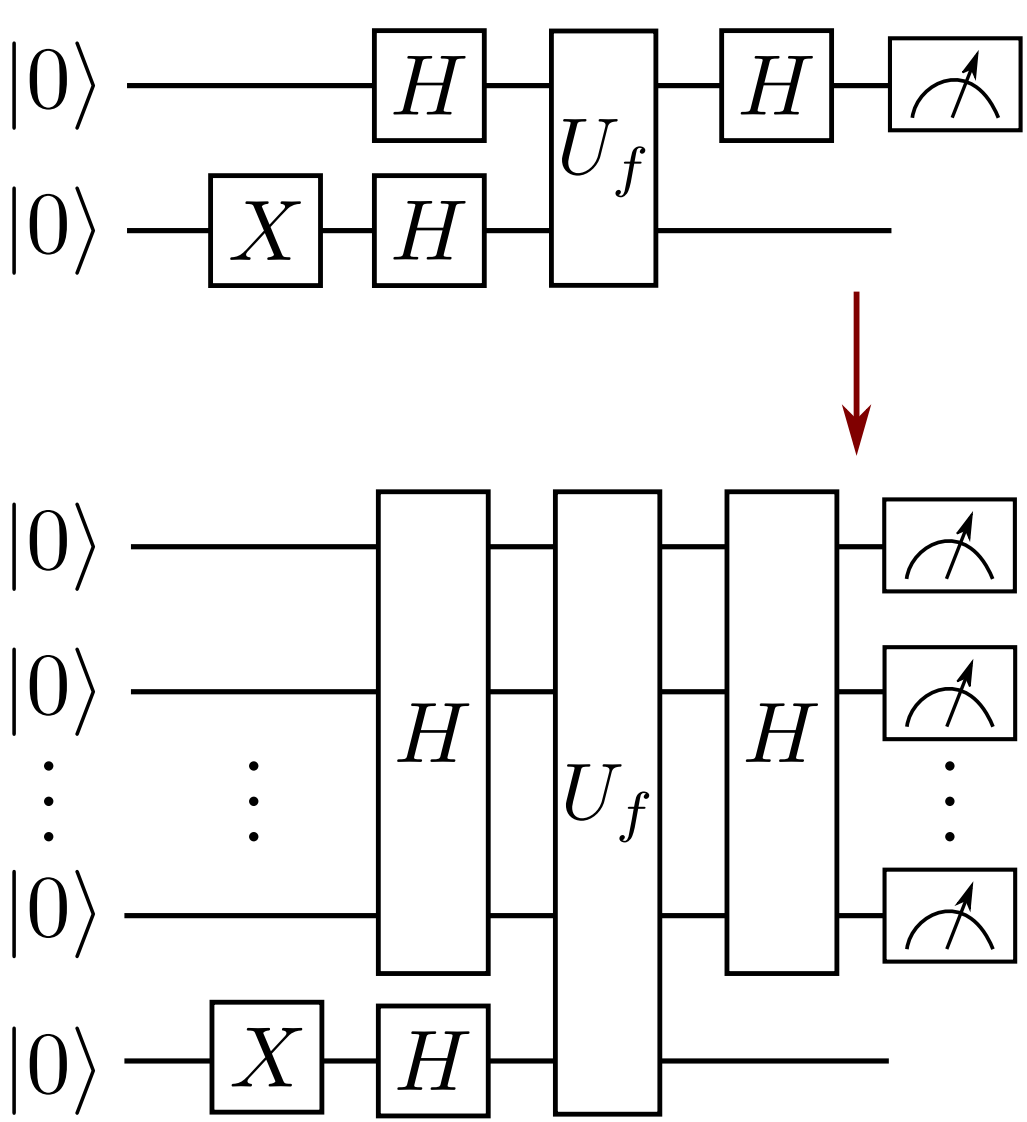
$$U_{f_4}|0\rangle|0\rangle = |0\rangle|0 \oplus f_4(0)\rangle =$$
$$U_{f_4}|0\rangle|1\rangle = |0\rangle|1 \oplus f_4(0)\rangle =$$
$$U_{f_4}|1\rangle|0\rangle = |1\rangle|0 \oplus f_4(1)\rangle =$$
$$U_{f_4}|1\rangle|1\rangle = |1\rangle|1 \oplus f_4(1)\rangle =$$

Deutsch's algorithm



Let's go back and implement this...

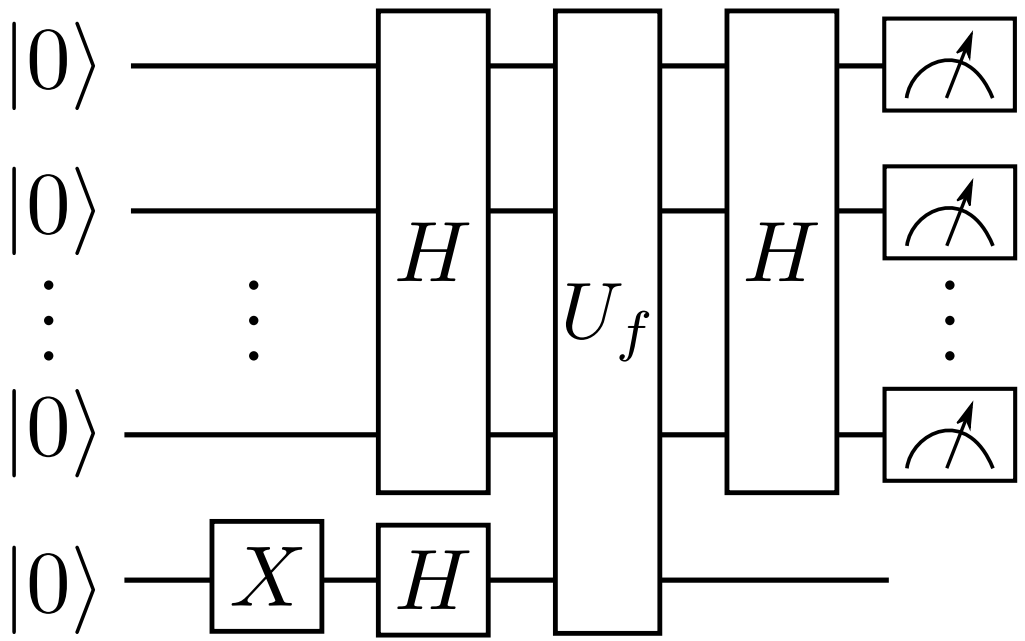
Generalization: Deutsch-Jozsa algorithm



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

Generalization: Deutsch-Jozsa algorithm

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



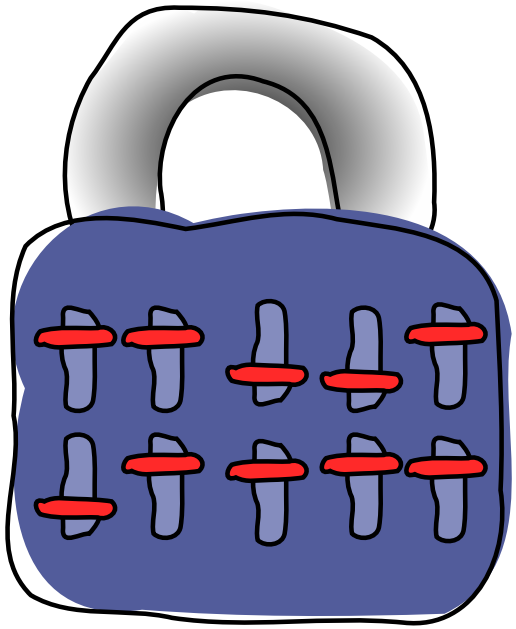
$|00\cdots0\rangle$ if f is constant

anything but $|00\cdots0\rangle$ if balanced

Grover's algorithm

Motivation

Let's break that lock!

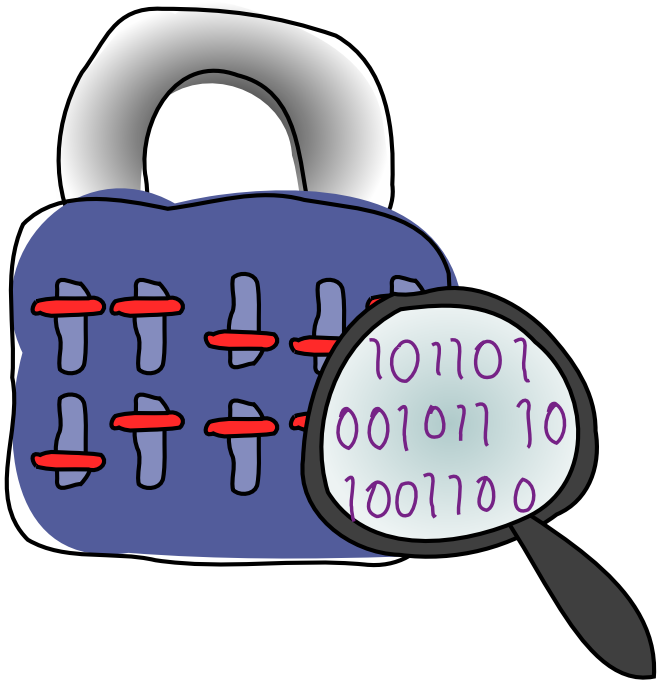


We input the combination to the lock as an n-bit (binary) string. The correct combination is labelled $\mathbf{s}$.

Image credit: Codebook node A.1

Motivation

How many times do we have to query the oracle to find the solution?



Classically: in the worst case we have to query the oracle times!

Grover's quantum search algorithm

The idea behind Grover's search algorithm is to start with a uniform superposition and then *amplify* the amplitude of the state corresponding to the solution.

In other words we want to go from the uniform superposition

to something that looks more like this:

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!

How do we do this?

Grover's algorithm: amplitude visualization

We start with the uniform superposition (i.e., perform a Hadamard transform on the qubits).

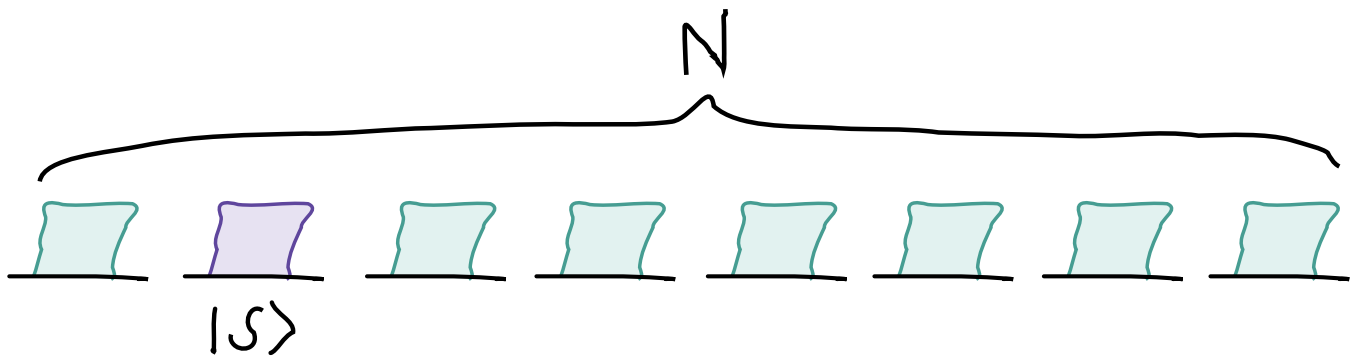


Image credit: Codebook node G.1

Grover's algorithm: amplitude visualization

If we apply the oracle, we flip the sign of the amplitude of the solution state:

$$|\mathbf{x}\rangle \to (-1)^{f(\mathbf{x})}|\mathbf{x}\rangle$$

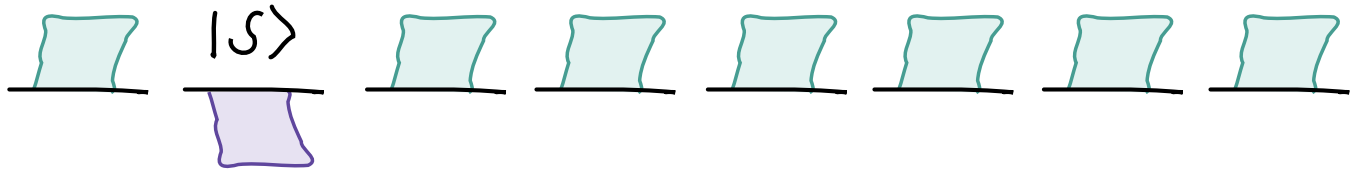


Image credit: Codebook node G.1

Grover's algorithm: amplitude visualization

Now what? Can't just apply the oracle again... need to do something different.

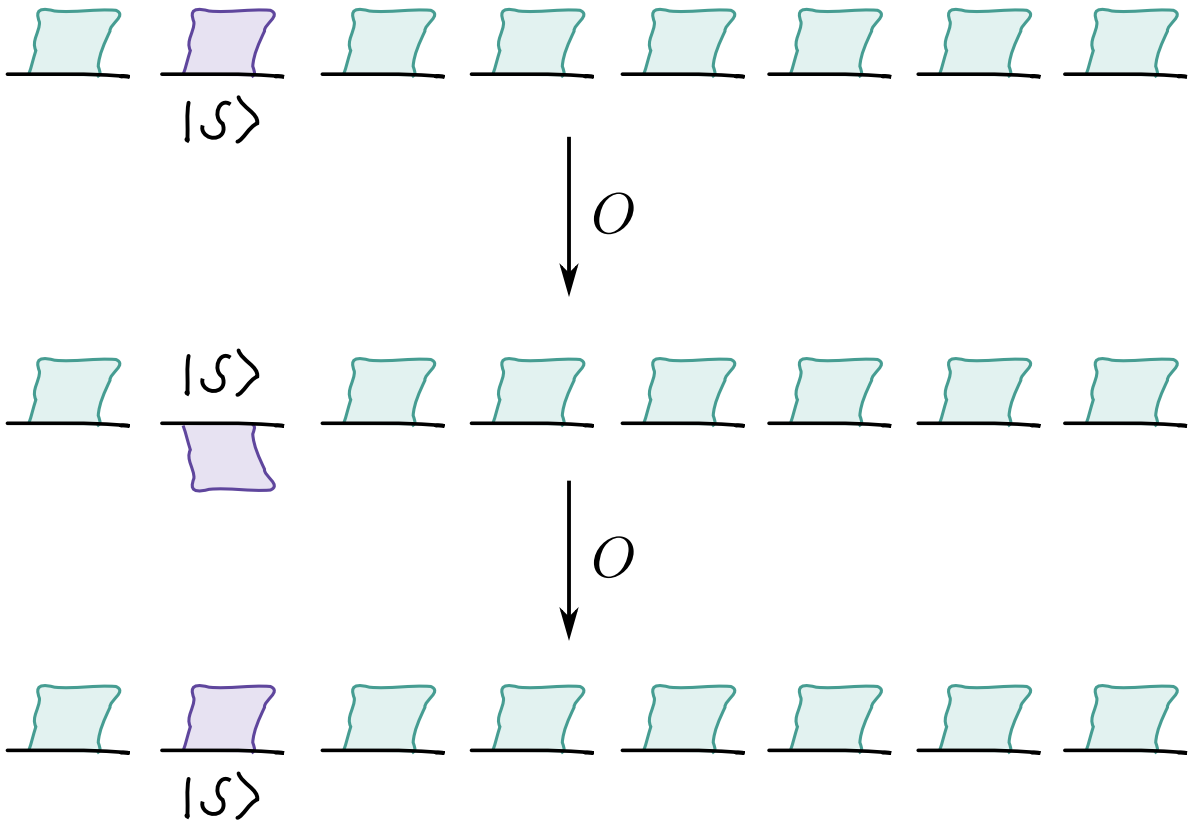
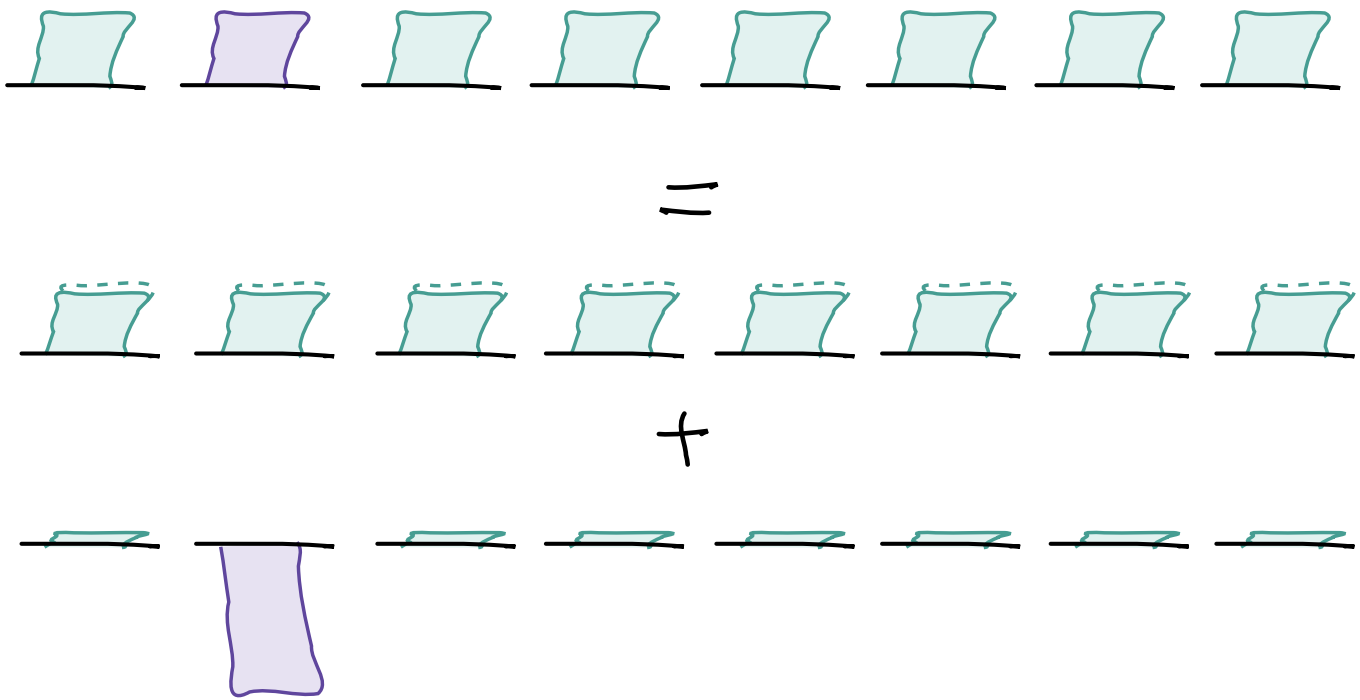


Image credit: Codebook node G.1

Grover's algorithm: amplitude visualization

Let's write the amplitudes in a different way:

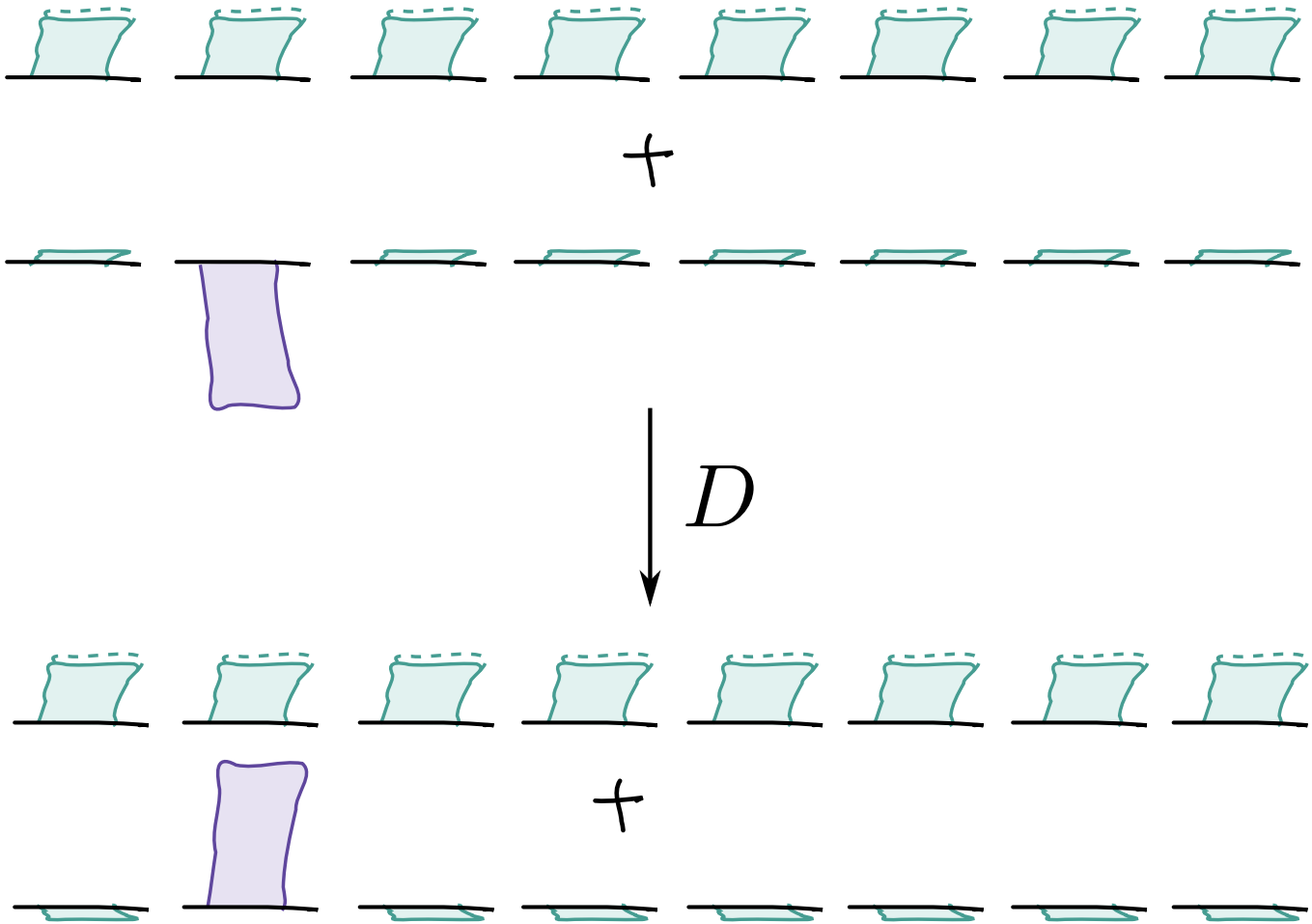


Why does this help?

Image credit: Codebook node G.1

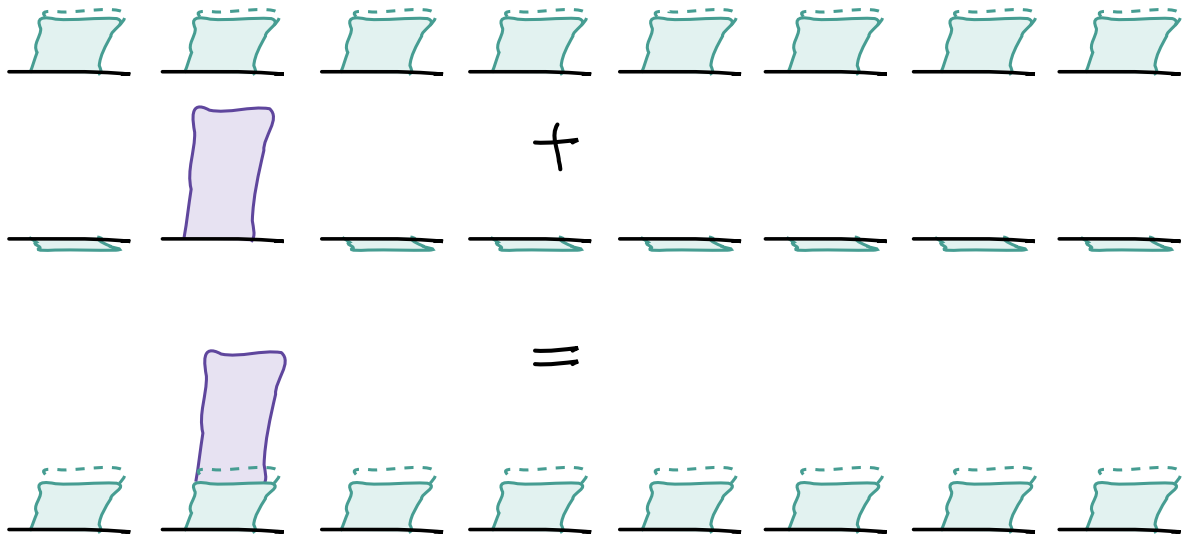
Grover's algorithm: amplitude visualization

What if we had an operation that would flip everything in the second part of the linear combination?



Grover's algorithm: amplitude visualization

Let's add these back together...



We have "stolen" some amplitude from the other states, and added it to the solution state!

Image credit: Codebook node G.1

Grover's algorithm: amplitude visualization

Doing this sequence once is one "iteration":

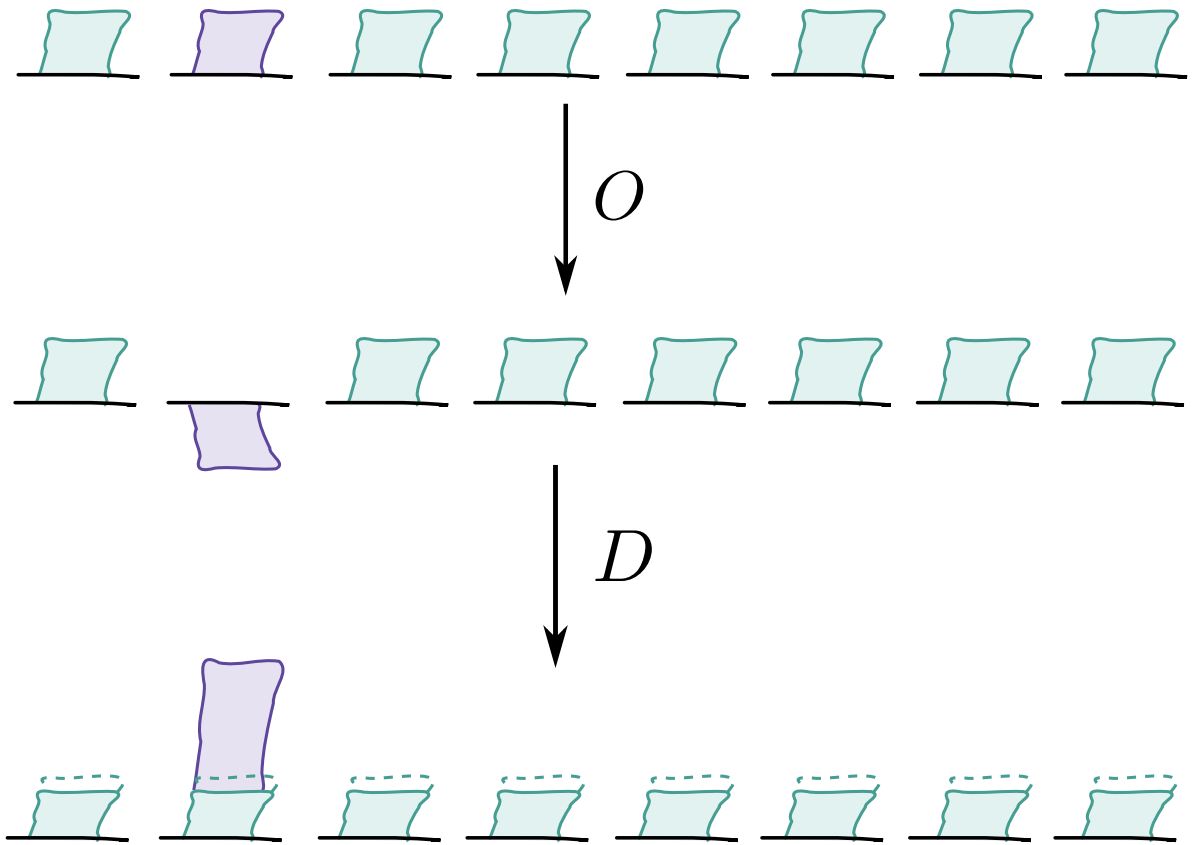
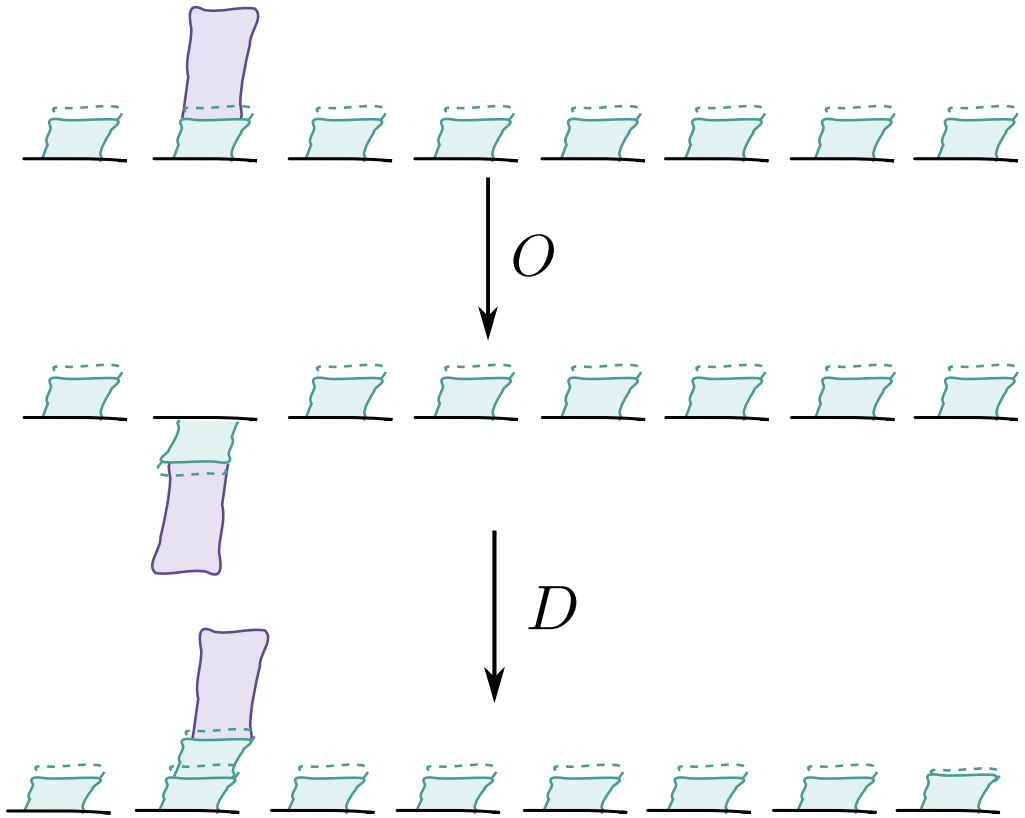


Image credit: Codebook node G.1

Grover's algorithm: amplitude visualization

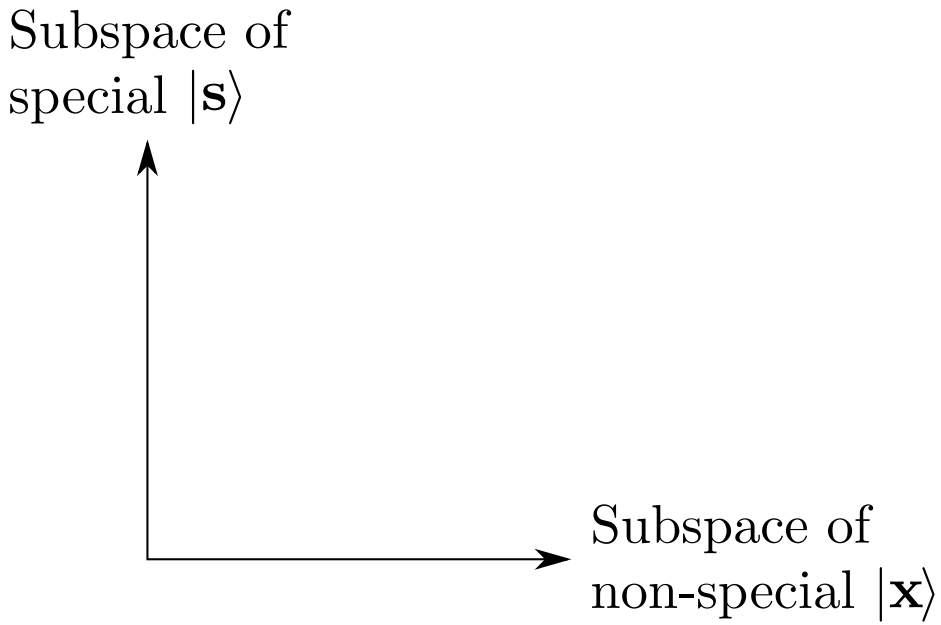
If we do it again, we can steal even more amplitude!



Grover's algorithm works by iterating this sequence multiple times until the probability of observing the solution state is maximized.

Image credit: Codebook node G.1

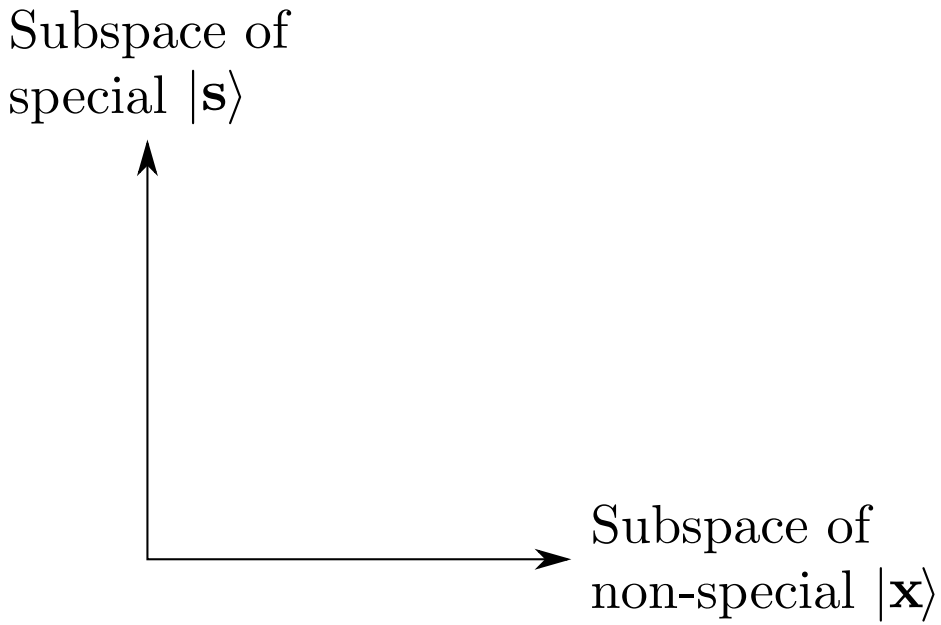
Grover's algorithm: geometric visualization



We can partition the subspace of all 2^n computational basis states into two parts:

1. The special state $|\mathbf{s}\rangle$

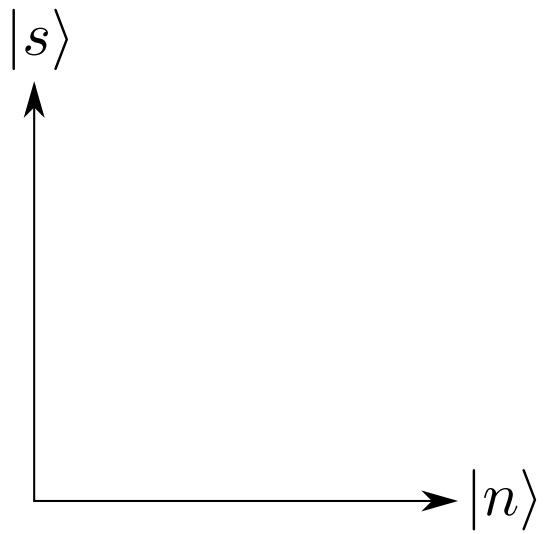
Grover's algorithm: geometric visualization



We can partition the subspace of all 2^n computational basis states into two parts:

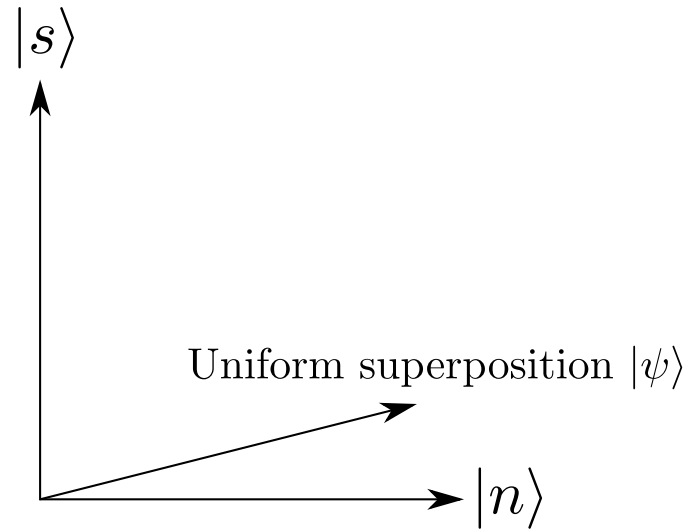
1. The special state $|\mathbf{s}\rangle$
2. All the other states

Grover's algorithm: geometric visualization



Let's write these out as superpositions:

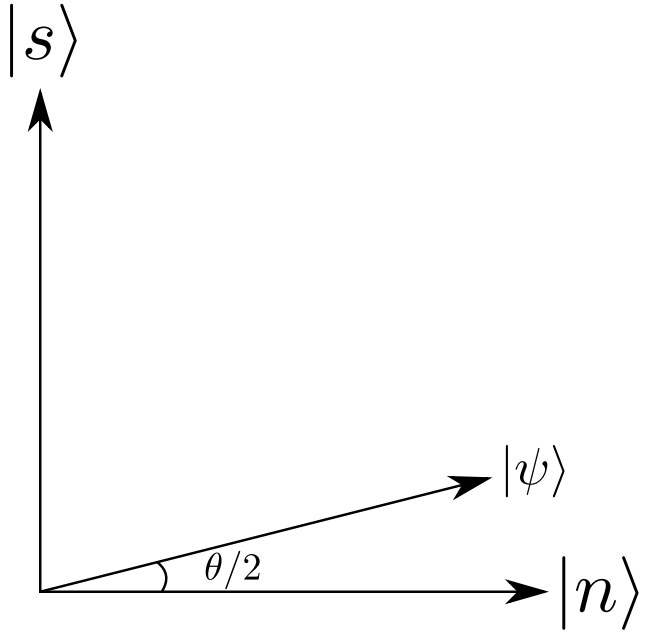
Grover's algorithm: geometric visualization



$$
\begin{aligned}
|s\rangle &= |\mathbf{s}\rangle \\
|n\rangle &= \frac{1}{\sqrt{2^n - 1}} \sum_{\mathbf{x} \neq \mathbf{s}} |\mathbf{x}\rangle
\end{aligned}
$$

We can write the uniform superposition in terms of these subspaces:

Grover's algorithm: geometric visualization

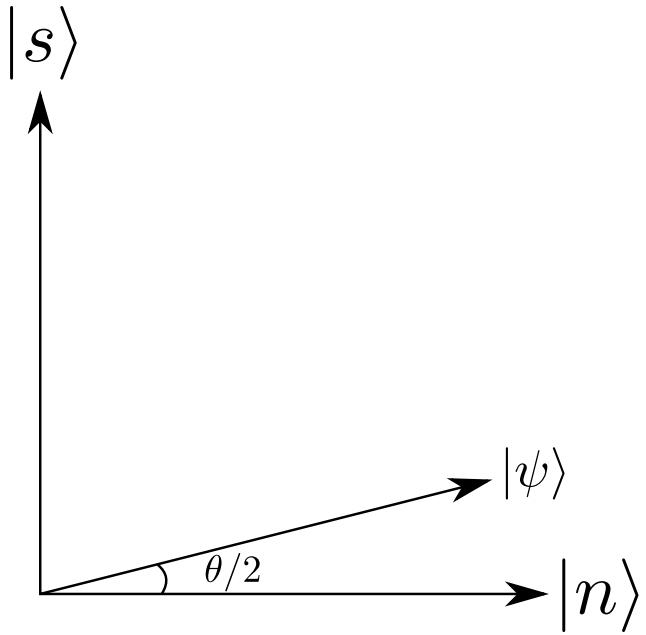


Instead of working with these complicated coeffients:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}}|s\rangle + \frac{\sqrt{2^n - 1}}{\sqrt{2^n}}|n\rangle,$$

let's rexpress them in terms of an angle θ:

Grover's algorithm: geometric visualization

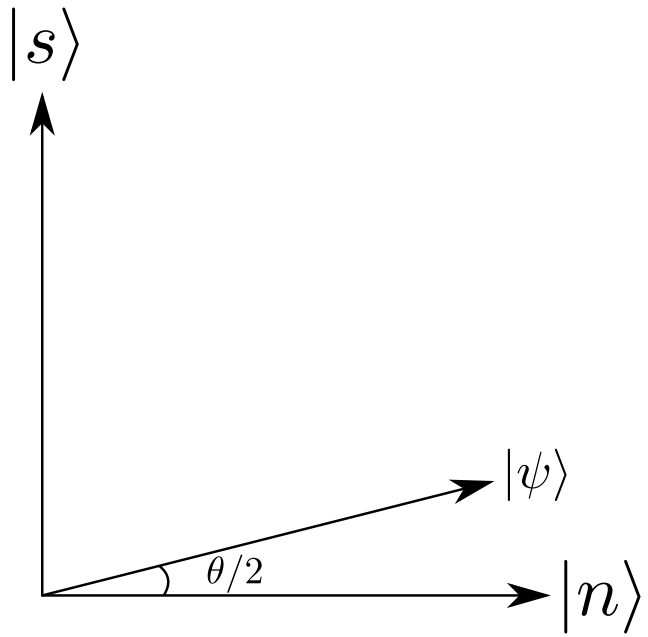


Now we want to apply some operations to this state

$$|\psi\rangle = \sin\left(\frac{\theta}{2}\right)|s\rangle + \cos\left(\frac{\theta}{2}\right)|n\rangle$$

in order to increase the amplitude of $|s\rangle$ while decreasing the amplitude of $|n\rangle$.

Grover's algorithm: geometric visualization

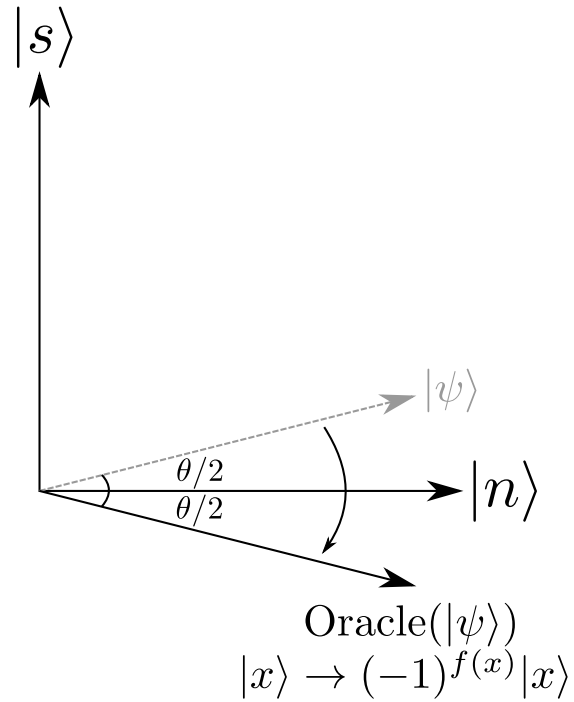


We will do this in two steps:

1. Apply the oracle O to 'pick out' the solution
2. Apply a 'diffusion operator' D to adjust the amplitudes.

We will call the product $DO = G$ the *Grover iteration*. Grover's algorithm consists of repeating this procedure many times.

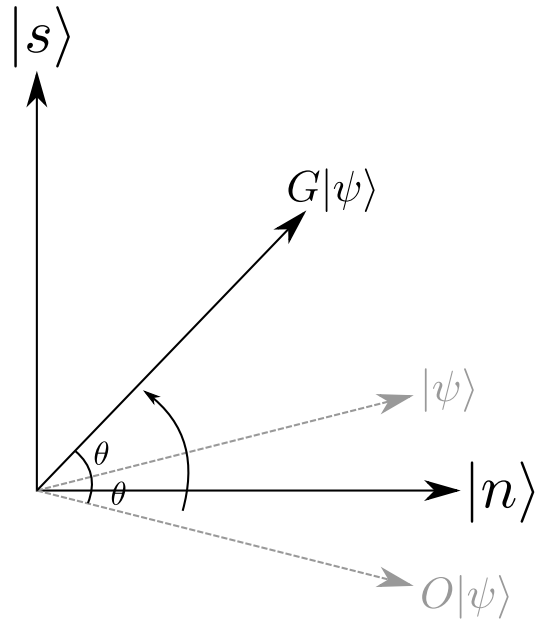
Grover's algorithm: geometric visualization



The effect of the oracle, $O|\psi\rangle$ *flips* the amplitudes of the basis states that are special.

We can visualize this as a *reflection about the subspace* of non-special elements.

Grover's algorithm: geometric visualization

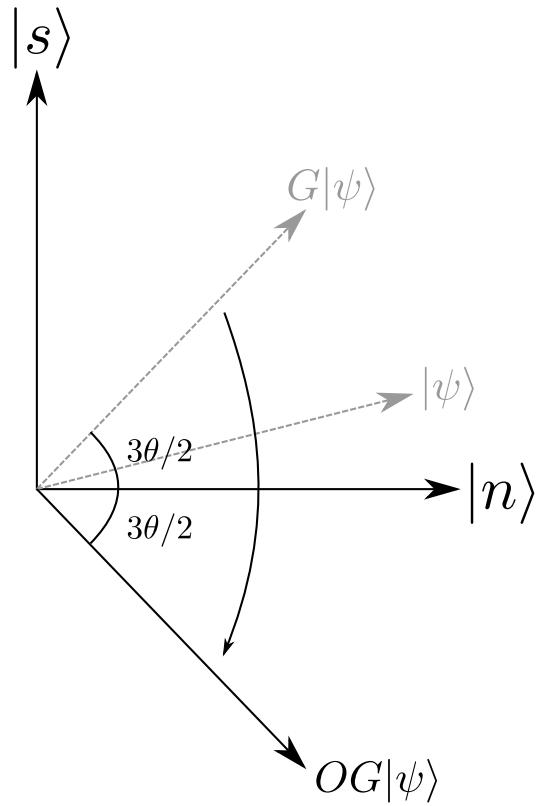


The diffusion operator is a bit less intuitive to interpret - it performs a *reflection about the uniform superposition* state.

A full Grover iteration $G = DO$ sends

$$G\left(\sin\left(\frac{\theta}{2}\right)|s\rangle + \cos\left(\frac{\theta}{2}\right)|n\rangle\right) =$$

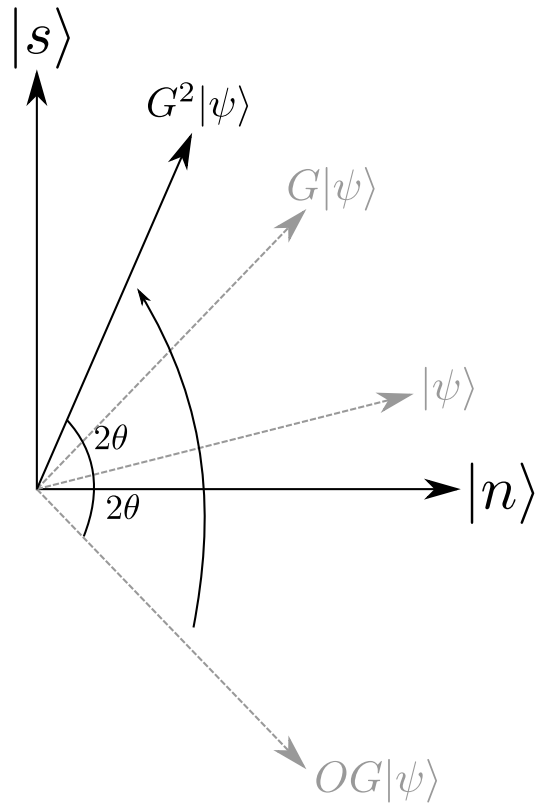
Grover's algorithm: geometric visualization



Now we repeat this...

Apply the oracle and reflect about the non-special elements.

Grover's algorithm: geometric visualization

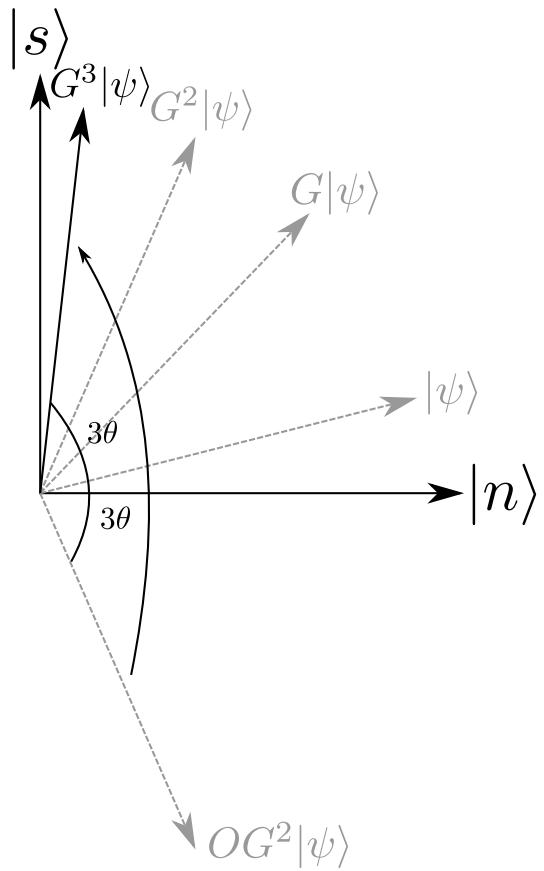


Apply the diffusion operator and reflect about the uniform superposition to boost the amplitude of the special state.

Grover's algorithm: geometric visualization

After k Grover iterations we will have the state

$$G^k|\psi\rangle =$$



It *is* possible to over-rotate! We can differentiate to find the optimal k:

$$k \leq \left\lceil \frac{\pi}{4}\sqrt{2^n} \right\rceil$$

After k operations we will be most likely to obtain the special state as our measurement outcome.

Performance of Grover's algorithm

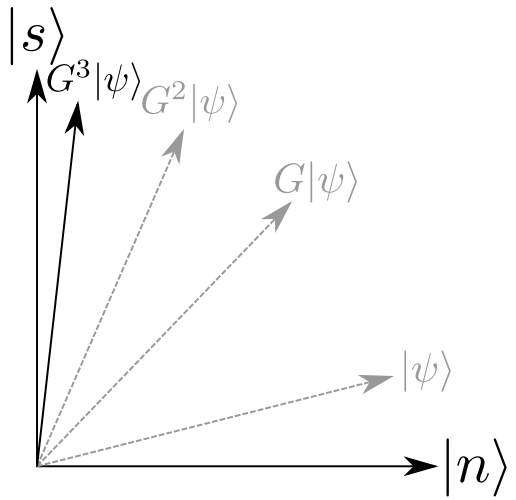
Recall that to solve the classical problem we needed to make 2^n oracle queries.

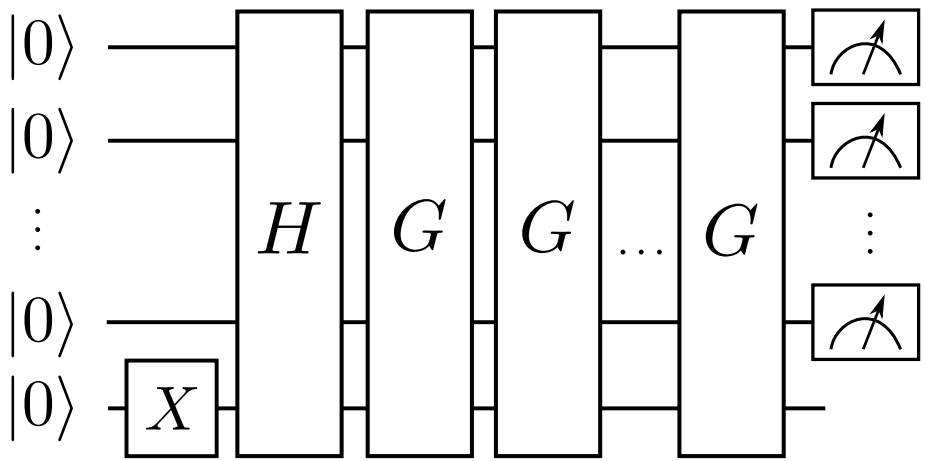
Using Grover's algorithm, we only need to make on the order of $\sqrt{2^n}$ queries!

Grover's algorithm provides a *polynomial speedup*, or *square-root* speedup over classical algorithms for searching unsorted spaces.

Implementing Grover search

The qubits all start in a uniform superposition, except for an auxiliary qubit in $|-\rangle$ to help with phase kickback. Then we repeatedly apply G which consists of applying the oracle, then the diffusion operator.





What do circuits for the oracle and diffusion look like?

The oracle circuit

Recall the function our oracle should implement for the problem of breaking a lock with a correct combination $\mathbf{s}$:

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

The oracle for this should do:

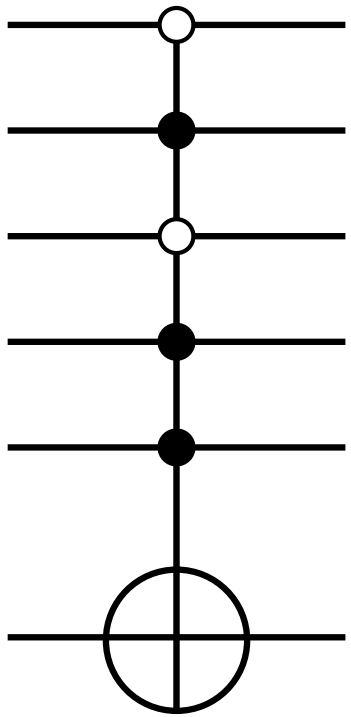
$$U|\mathbf{x}\rangle|y\rangle = |\mathbf{x}\rangle|y \oplus f(\mathbf{x})\rangle$$

When $|y\rangle$ is set to the $|-\rangle$ state, a phase will be kicked back and we flip the sign of the amplitude.

There is only one input $\mathbf{x}$ for which this we need to add 1 to the other qubit...

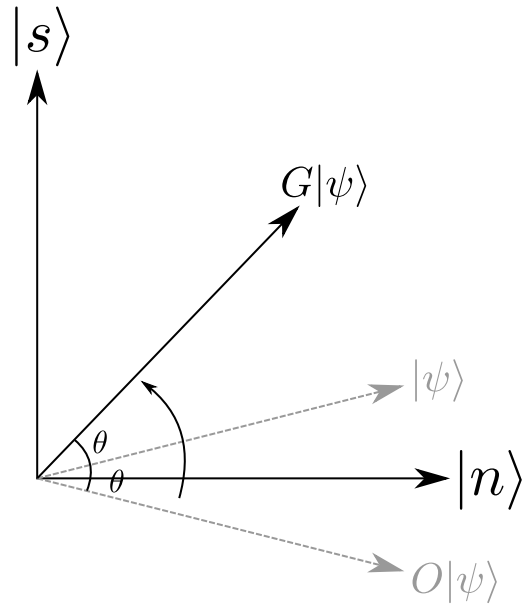
The oracle circuit

We can use a controlled X gate, where the state of the control qubits matches the solution state.



The diffusion circuit

The diffusion operator performs a reflection about the uniform superposition state.

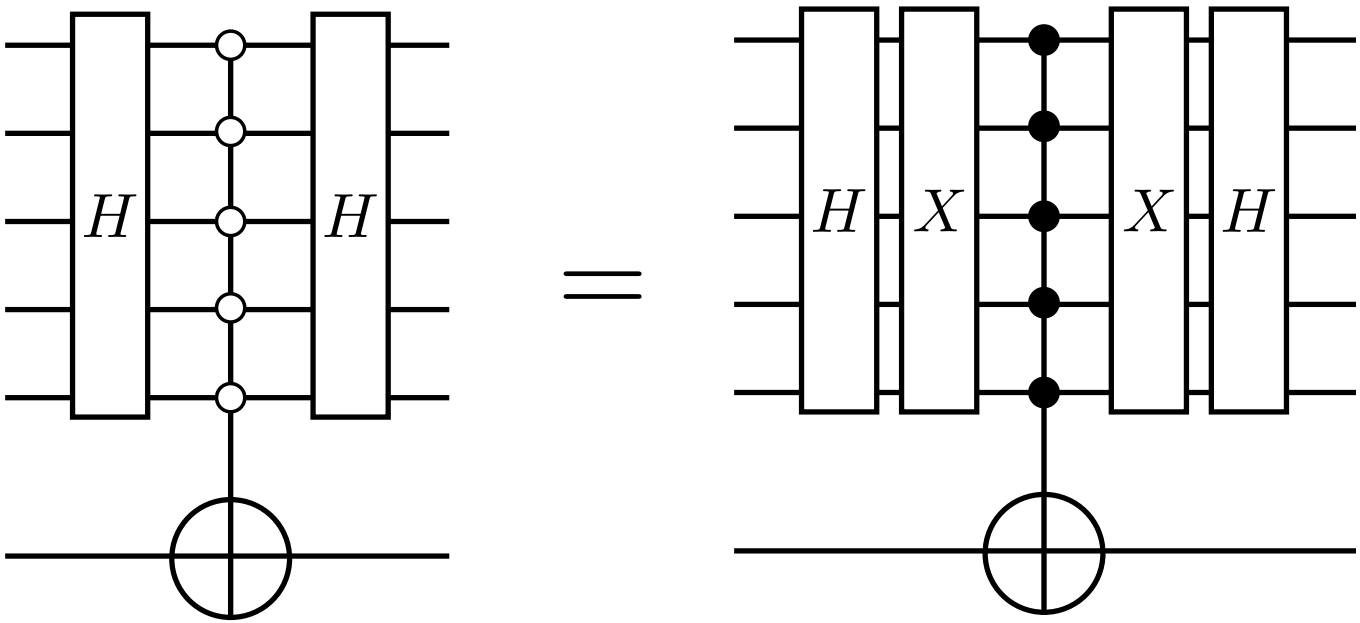


Recall that the uniform superposition is

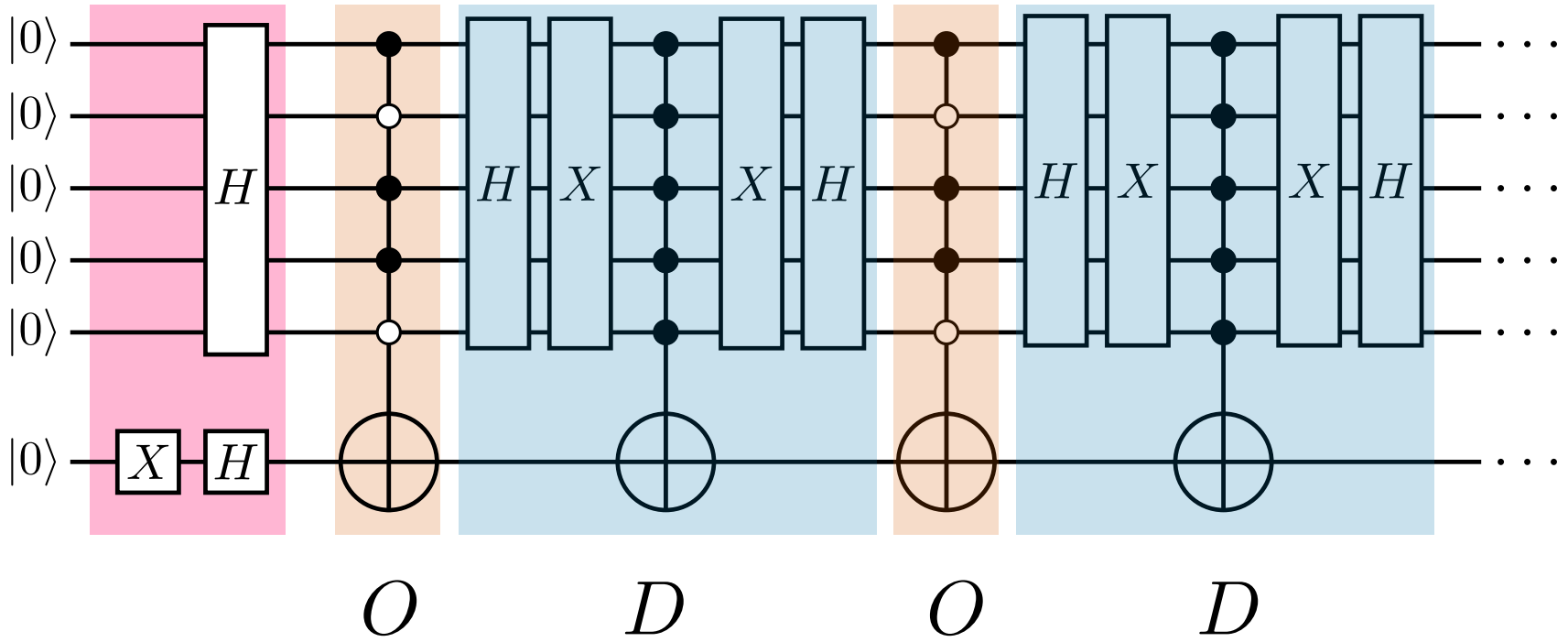
$$|\psi\rangle = (H \otimes H \otimes \cdots \otimes H)|00\cdots0\rangle = H^{\otimes n}|0\rangle^{\otimes n}$$

The diffusion circuit

We can implement the reflection by first applying a Hadamard to change to the computational basis; performing a reflection around the equivalent state; changing the basis back.



The full Grover circuit



After iterating this the correct number of times, the most likely measurement outcome of the top set of qubits should correspond to the solution state. Let's try it...

Next time

Content:

- Overview of quantum transforms and compilation

Action items:

1. Get started on Assignment 2 (can do the first 3 problems now)

Recommended reading:

- Codebook nodes G.1-G.5
- Nielsen & Chuang 6.1