CPEN 400Q Lecture 09
The oracle, query complexity, and Deutsch's algorithm

Monday 5 February 2024

Announcements

- No quiz today
- Project details later this week
- First literacy assignment and A2 available tomorrow (both are short)

Module 2 learning outcomes

Learning outcomes:

- explain what it means for an algorithm to have a quantum speedup
- define quantum oracles and query complexity
- implement oracles and Grover's algorithm in PennyLane
- identify the different components of the quantum compilation stack
- define and list common universal gate sets
- estimate the resources required to run a quantum algorithm
- implement quantum transforms to perform simple circuit optimization in PennyLane

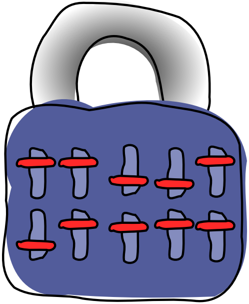
Today

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

Oracles: motivating problem

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



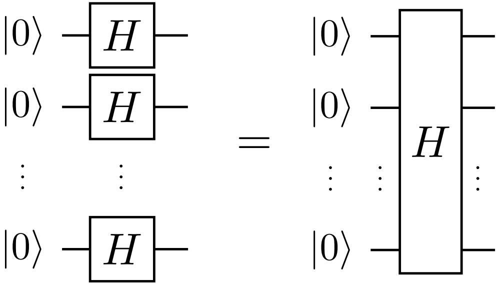
How do we solve this classically?

Image credit: Codebook node A.1

Idea: use superposition

Can we do better with a quantum computer?

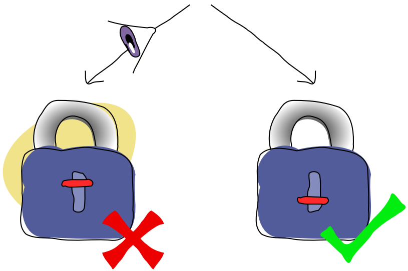
What if we take n qubits and put them in a superposition with all possible combinations?



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.

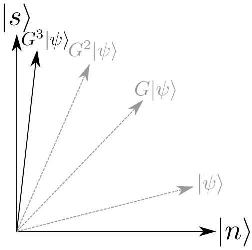


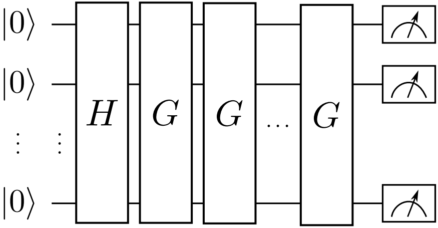
Quantum computers are **NOT** faster because they can “compute everything at the same time.”

Image credit: Codebook node A.1

Can we solve this problem better with a quantum computer?

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



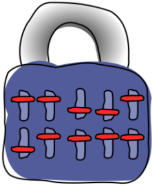


We will explore the algorithmic primitives that are involved, and some other cases where we can do better with quantum computing.

Oracles

Motivating problem

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



Classically, we would have to try every possible combination. If there are n bits, that's 2^n possible tries. Can we do better with a quantum computer?

Image credit: Codebook node A.1

6 / 26

What is a “try”?

Oracles

Let

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

We can represent a "try" as a function:

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**.

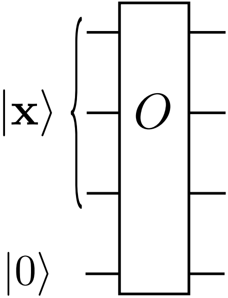
We will need quantum operations to play the role of the oracle.

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

Exercise: Consider a 2-qubit system where $f(01) = 1$, and $f(\mathbf{x}) = 0$ for all other $\mathbf{x}$. What is the action of the oracle?

Input state	Output state
$\lvert 00\rangle\lvert 0\rangle$	
$\lvert 01\rangle\lvert 0\rangle$	
$\lvert 10\rangle\lvert 0\rangle$	
$\lvert 11\rangle\lvert 0\rangle$	

Quantum oracles



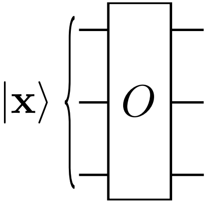
$$
\begin{aligned}
O|000\rangle|0\rangle &= |000\rangle|0\rangle\\
O|001\rangle|0\rangle &= |001\rangle|0\rangle\\
O|010\rangle|0\rangle &= |010\rangle|0\rangle\\
O|011\rangle|0\rangle &= |011\rangle|0\rangle\\
O|100\rangle|0\rangle &= |100\rangle|0\rangle\\
O|101\rangle|0\rangle &= |101\rangle|0\rangle\\
O|110\rangle|0\rangle &= |110\rangle|1\rangle\\
O|111\rangle|0\rangle &= |111\rangle|0\rangle
\end{aligned}
$$

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

Exercise: Consider a 2-qubit system where $f(11) = 1$, and $f(\mathbf{x}) = 0$ for all other $\mathbf{x}$. What is the action of the oracle?

Input state	Output state
$\vert 00\rangle$	
$\vert 01\rangle$	
$\vert 10\rangle$	
$\vert 11\rangle$	

Quantum oracles



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

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	$f_1(0) = 0$ $f_1(1) = 0$	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* (same output no matter what the input), and f_3 and f_4 are balanced.

How many **classical** 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	$f_1(0) = 0$ $f_1(1) = 0$	f_2	$f_2(0) = 1$ $f_2(1) = 1$
f_3	$f_3(0) = 0$ $f_3(1) = 1$	f_4	$f_4(0) = 1$ $f_4(1) = 0$

How many **quantum** 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	$f_1(0) = 0$ $f_1(1) = 0$	f_2	$f_2(0) = 1$ $f_2(1) = 1$
f_3	$f_3(0) = 0$ $f_3(1) = 1$	f_4	$f_4(0) = 1$ $f_4(1) = 0$

The secret relies on *phase kickback*.

Exercise: what happens when we apply a CNOT to these 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 can write a general version of this effect:

How does this relate to finding if a function is constant / balanced?

Suppose we have a black box (oracle), U_f, that implements any of these four functions, f:

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

Initializing the second qubit to $|-\rangle$ will allow us to learn the value of $f(0) \oplus f(1)$ with a single query.

Exercise: why $f(0) \oplus f(1)$?

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

$$=$$

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

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

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

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

Remember how we generalized the result for CNOT:

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

So we can write

Exercise: CNOT acts like U_f for one specific $f(x)$. Which one?

How to use this to get $f(0) \oplus f(1)$?

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

$$=$$

$$=$$

Deutsch's algorithm

$=$

$=$

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

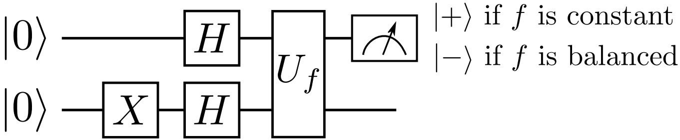
If the function is constant, $f(0) \oplus f(1) = 0$ and the state is

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

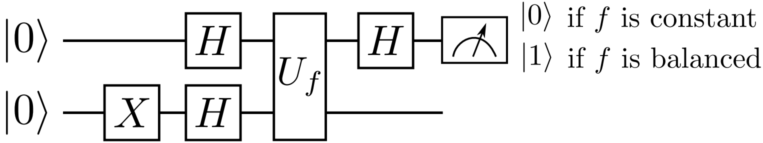
But if the function is balanced, $f(0) \oplus f(1) = 1$ and the state is

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

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

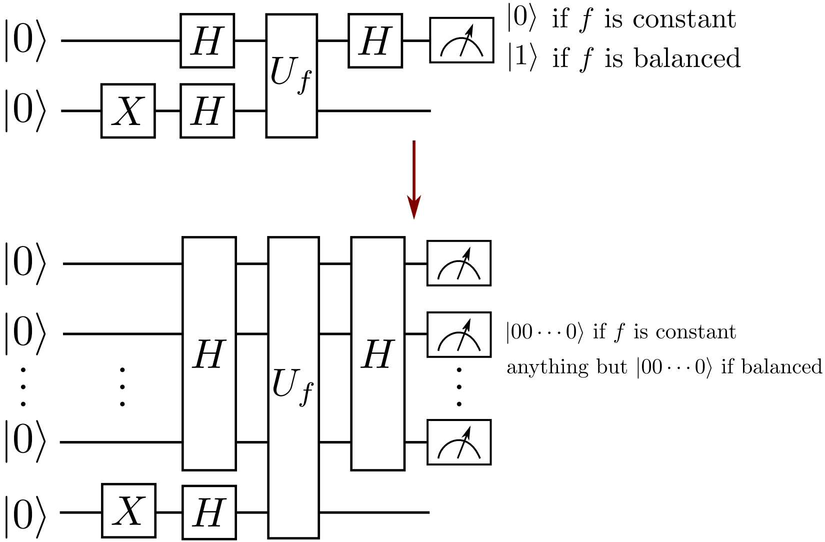


Or equivalently,

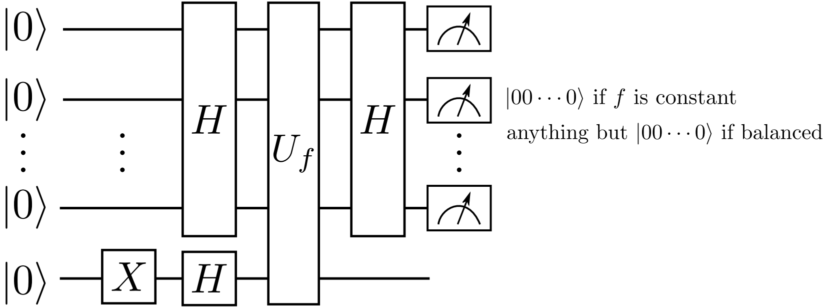


We call U_f just once, but obtain information about the relationship between $f(0)$ and $f(1)$! Let's implement it.

Generalization: Deutsch-Jozsa algorithm



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



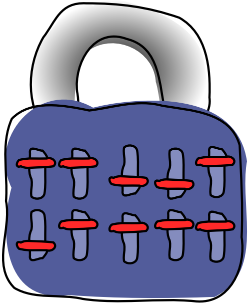
(Challenge: try implementing it yourself to check if this works!)

A few other interesting algorithms:

Bernstein-Vazirani algorithm (will see on A2)
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

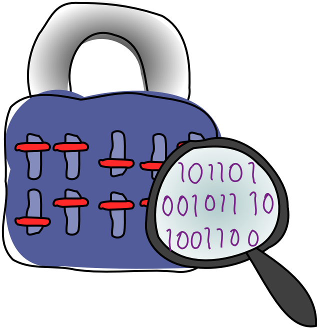
Let's break that lock!



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

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



Classical: in the worst case, times

Quantum: 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

to something that looks more like this:

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

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

Next time

Content:

- Amplitude amplification and Grover's algorithm

Action items:

1. Start thinking about project groups
2. Assignment 2 / literacy assignment coming later this week

Recommended reading:

- Codebook modules A and G