CPEN 400Q Lecture 08
The oracle, query complexity, and Deutsch's algorithm

Monday 3 February 2025

Announcements

- No quiz today; no tutorial tomorrow
- Midterms grades posted and tests available for pickup on *Thursday afternoon*
  - you'll be informed after class today if you need to do make-up oral (will schedule for *next week*)
- Project details coming next week (start thinking about groups)
- First literacy assignment and A2 available soon

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
- perform simple circuit optimizations in PennyLane

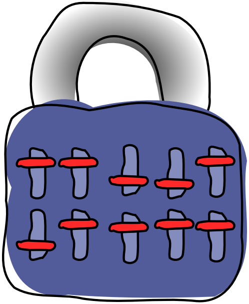
Today

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

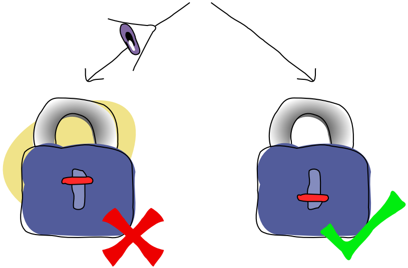
Can we do better with a quantum computer?

Idea: take n qubits and construct a *uniform* superposition of all combinations

Exercise: design a circuit to create this state.

Idea: use superposition

Measurements are probabilistic - the solution just being "in" the superposition doesn't make it easier to solve the problem.



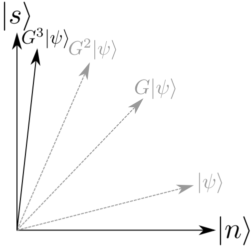
Quantum computers are **NOT** faster because they "compute everything at the same time."

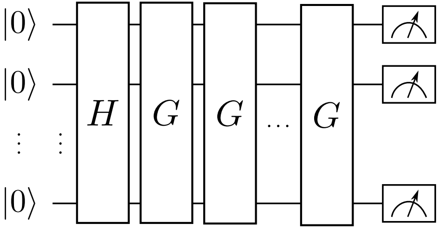
Image credit: Codebook node A.1

Solving problems with quantum computers

Can we solve this problem better with a quantum computer?

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





We will explore the algorithmic primitives, and see other cases where quantum computers have a **speed up in query complexity**.

Oracles

Revisit the lock problem and model a "try" as a function. Let

- $\mathbf{x}$ be an n-bit string (an input combination)
- $\mathbf{s}$ be the correct combination

Don't care *how* $f(\mathbf{x})$ is evaluated, only that it gives a "yes/no" answer. $f(\mathbf{x})$ is a black box, or **oracle**.

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

Must determine

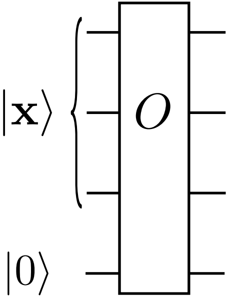
- a mathematical description of an oracle
- where to record the output, i.e., the value of $f(\mathbf{x})$

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

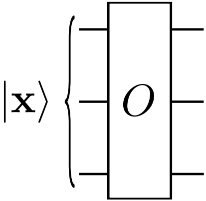


$$
\begin{aligned}
O|000\rangle|0\rangle &= |000\rangle|0\rangle \\
O|001\rangle|0\rangle &= |001\rangle|0\rangle \\
O|010\rangle|0\rangle &= |010\rangle|0\rangle \\
O|011\rangle|0\rangle &= |011\rangle|0\rangle \\
O|100\rangle|0\rangle &= |100\rangle|0\rangle \\
O|101\rangle|0\rangle &= |101\rangle|0\rangle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

Idea 2: encode the result in the phase

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
$\|00\rangle$	
$\|01\rangle$	
$\|10\rangle$	
$\|11\rangle$	



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

Motivation: You have access to an oracle that implements one of the following 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*, f_3 and f_4 are *balanced*.

How many **classical** oracle queries are needed to determine if the oracle function is constant or balanced?

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** oracle queries are needed to determine if the oracle function is constant or balanced?

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

Phase kickback

The secret is the *phase kickback* trick.

Exercise: apply CNOT_{01} to the following two-qubit states.

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

Deutsch's algorithm

Let U_f be an oracle for one of the four functions:

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

Initializing the second qubit to $|-\rangle$ allows us to learn $f(0) \oplus f(1)$ with a single oracle query.

Q: How does $f(0) \oplus f(1)$ relate to properties of the function?

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

$$=$$

If $f(x)=0$,

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

If $f(x)=1$,

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

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 $f_j(x)$. 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

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

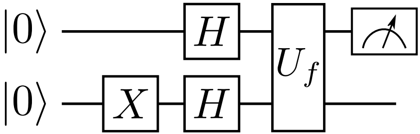
If the function is constant, $f(0) \oplus f(1) = 0$ and the state is

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

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

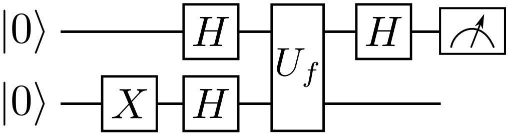
$$U_f \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) =$$

Deutsch's algorithm as a circuit:



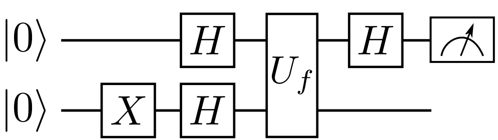
$|+\rangle$ if f is constant
$|-\rangle$ if f is balanced

Equivalently,

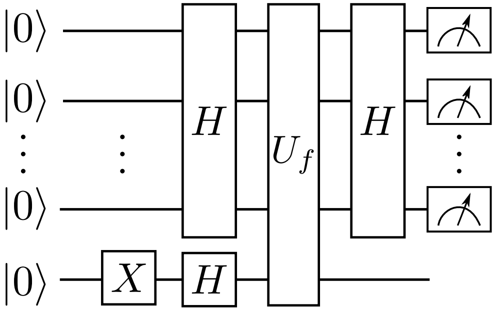


$|0\rangle$ if f is constant
$|1\rangle$ if f is balanced

Generalization: Deutsch-Jozsa algorithm

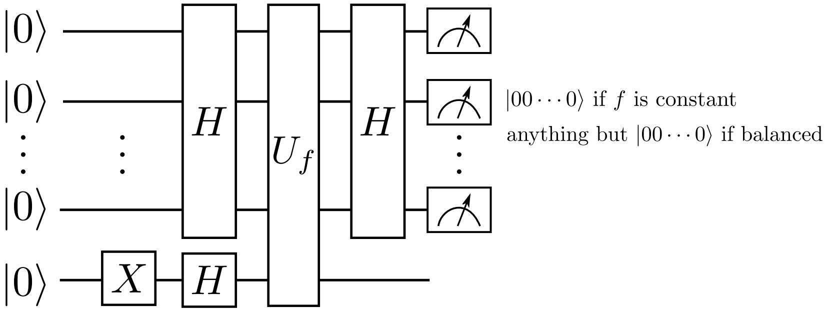


$|0\rangle$ if f is constant
$|1\rangle$ if f is balanced



$|00\cdots 0\rangle$ if f is constant
anything but $|00\cdots 0\rangle$ if balanced

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



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

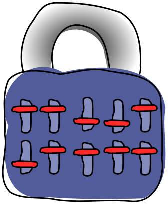
A few other interesting algorithms:

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 oracle queries.

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 number of oracle queries.

Let's break that lock!

Input combination is an n-bit string. Correct combination is $\mathbf{s}$.
How many oracle queries the oracle to find the solution?



Classical:

Quantum:

Image credit: Codebook node A.1

Start with a uniform superposition, then *amplify* the amplitude of the solution state $|\mathbf{s}\rangle$.

In other words, go from the uniform superposition

to something that looks like:

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

Next time

Content:

- Amplitude amplification and Grover's algorithm

Action items:

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

Recommended reading:

- For today: Codebook module BA; Nielsen & Chuang 1.4.1-1.4.4
- For next class: Codebook module GA; Nielsen & Chuang 6.1