LESSON 5.1
DEUTSCH'S ALGORITHM

Valter Uotila

OUTLINE

Introduction

Oracles

Deutsch's algorithm

Demonstrations

RECAP WHAT WE HAVE LEARNED SO FAR

- Qubits $|0\rangle$, $|1\rangle$, quantum logic gates, superposition and engtanglement
- Their implementation in Qiskit
- This presentation is based on [1, 2] and the tutorial on IBM quantum lab

RECAP WHAT WE HAVE LEARNED SO FAR

For this presentation, it is useful to recall Hadamard-gate:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}.$$

and

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

RECAP WHAT WE HAVE LEARNED SO FAR

Also, it is good to recall that Hadamard-gate is its own inverse

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

and

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

MOTIVATION & BACKGROUND

- Main challenge in quantum computing: develop algorithms that are provably faster than corresponding classical algorithms

MOTIVATION & BACKGROUND

- Main challenge in quantum computing: develop algorithms that are provably faster than corresponding classical algorithms
- This has appeared to be hard!

MOTIVATION & BACKGROUND

- Main challenge in quantum computing: develop algorithms that are provably faster than corresponding classical algorithms
- This has appeared to be hard!
- Deutsch's algorithm (1985) is the most simple example where quantum computing is provably faster than any other classical algorithm

MOTIVATION & BACKGROUND

- Main challenge in quantum computing: develop algorithms that are provably faster than corresponding classical algorithms
- This has appeared to be hard!
- Deutsch's algorithm (1985) is the most simple example where quantum computing is provably faster than any other classical algorithm
- No practical usage but theoretically important

MOTIVATION & BACKGROUND

- Main challenge in quantum computing: develop algorithms that are provably faster than corresponding classical algorithms
- This has appeared to be hard!
- Deutsch's algorithm (1985) is the most simple example where quantum computing is provably faster than any other classical algorithm
- No practical usage but theoretically important
- This and the next lesson will be about this algorithm (2 qubits) and its generalization (Deutsch-Jozsa algorithm, $n > 2$ qubits)

DEUTSCH'S PROBLEM

Constant vs. balanced functions

Let $f\colon \{0,1\} \to \{0,1\}$ be a function. There are totally four different such functions:

1. $f(0) = 0$ and $f(1) = 0$ (constant 0 function)
2. $f(0) = 1$ and $f(1) = 1$ (constant 1 function)
3. $f(0) = 0$ and $f(1) = 1$ (identity function – does not change anything)
4. $f(0) = 1$ and $f(1) = 0$ (swap function – changes the bits)

DEUTSCH'S PROBLEM

1. Found out if f is constant or balanced

DEUTSCH'S PROBLEM

1. Found out if f is constant or balanced
2. Classically, any algorithm that determines if f is constant or balanced, needs two steps: we need to evaluate both $f(0)$ and $f(1)$

DEUTSCH'S PROBLEM

1. Found out if f is constant or balanced
2. Classically, any algorithm that determines if f is constant or balanced, needs two steps: we need to evaluate both $f(0)$ and $f(1)$
3. Quantum computer manages to solve the problem with just one step!

DEUTSCH'S PROBLEM

1. Found out if f is constant or balanced
2. Classically, any algorithm that determines if f is constant or balanced, needs two steps: we need to evaluate both $f(0)$ and $f(1)$
3. Quantum computer manages to solve the problem with just one step!
4. The algorithm that solves the problem is called Deutsch's algorithm

DEUTSCH'S PROBLEM MORE FORMALLY

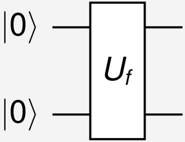
- Define a function g so that $g(f) = 0$ if f is constant and $g(f) = 1$ if f is balanced

DEUTSCH'S PROBLEM MORE FORMALLY

- Define a function g so that $g(f) = 0$ if f is constant and $g(f) = 1$ if f is balanced
- This is a decision problem

NOTATION FOR ORACLES IN CIRCUIT DIAGRAMS

Oracles have simple expression in circuit diagrams:



ENCODING THE FUNCTION

1. How do we express the function g in a quantum computer?

ENCODING THE FUNCTION

1. How do we express the function g in a quantum computer?
2. Solution: **oracle**

ENCODING THE FUNCTION

1. How do we express the function *g* in a quantum computer?
2. Solution: **oracle**
3. Recall that all the quantum gates need to be reversible. Oracles will be reversible and unitary as well.
4. In quantum computing, oracle is a simple matrix that works as oracles work in computer science: abstract black-box machine for decision problems.
5. Deutsch's algorithm speedup with oracle: any classical algorithm requires two access to the oracle whereas quantum computer requires just one!

ENCODING THE FUNCTION

Let x and y be binary variables. We define the function g with the following formula

$$g|xy\rangle = |x\rangle|y \oplus f(x)\rangle,$$

where

$$y \oplus f(x) = y + f(x) \mod 2$$

is the conventional notation for addition modulo 2. For example, if f is an identity, then

$$g|00\rangle = |0\rangle|0 \oplus f(0)\rangle = |0\rangle|0 \oplus 0\rangle = |00\rangle.$$

ENCODING THE FUNCTION

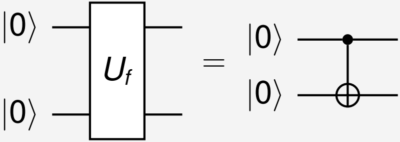
We can evaluate the function g in all the cases for all four functions f:

Case	1	2	3	4
g	$f(0,1)=(0,1)$	$f(0,1)=(1,0)$	$f(0,1)=0$	$f(0,1)=1$
$g\|00\rangle$	$\|00\rangle$	$\|01\rangle$	$\|00\rangle$	$\|01\rangle$
$g\|01\rangle$	$\|01\rangle$	$\|00\rangle$	$\|01\rangle$	$\|00\rangle$
$g\|10\rangle$	$\|11\rangle$	$\|10\rangle$	$\|10\rangle$	$\|11\rangle$
$g\|11\rangle$	$\|10\rangle$	$\|11\rangle$	$\|11\rangle$	$\|10\rangle$

CASE 1

In the case that f is identity, the oracle operates exactly as CNOT! Thus the oracle for this case is simply:

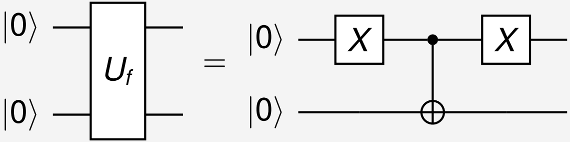
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



CASE 2

In the case that f is swap, the oracle operates exactly as not-CNOT:

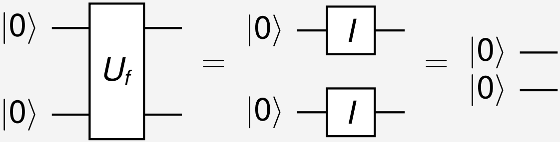
$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



CASE 3

In the case that f is constant 0, the oracle operates as identity:

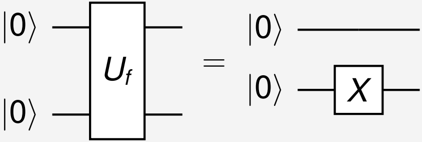
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



CASE 4

In the case that f is constant 1, the oracle applies not-gate to the second qubit:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



CODING ORACLES IN QISKIT AND OTHER FRAMEWORKS

We have at least two options to code oracles in implementations:

1. Create concrete matrix representation (for example with numpy) and transform that into a gate
2. Use X-gates and CNOTs

The previous slides showed examples of both of these

OUTLINE

1. Intuition behind the algorithm
2. Circuit that solves the problem
3. Mathematical solution
4. Qiskit implementation and running the algorihtm in IBM quantum systems

INITIAL SETTING

We are provided one of the four oracles in the previous slides. We do not know if the oracle encodes a constant or balanced function f. Now we want to find out which one it is. Classically this requires two oracle calls, but now we solve the problem with just one oracle call!

INTUITION: UNIFORM SUPERPOSITION

1. Classically, we need to evaluate the oracle for both 0 and 1

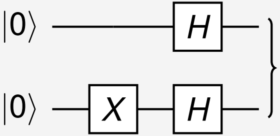
INTUITION: UNIFORM SUPERPOSITION

1. Classically, we need to evaluate the oracle for both 0 and 1
2. Quantumly, we create an equal superposition state of 0 and 1 using *Hadamard transform* and evaluate the oracle for the state. This is a kind of parallel execution

INTUITION: UNIFORM SUPERPOSITION

1. Classically, we need to evaluate the oracle for both 0 and 1
2. Quantumly, we create an equal superposition state of 0 and 1 using *Hadamard transform* and evaluate the oracle for the state. This is a kind of parallel execution
3. We apply Hadamard transform again
4. Finally, we measure the outcome. If we obtain 0, we know that the function is constant. If we obtain 1, the function is balanced.
5. The exact reason why the algorithm works is convenient to see mathematically

HADAMARD TRANSFORM



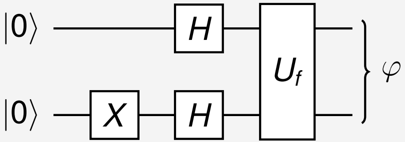
$$\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$$

MATHEMATICALLY

Precisly,

$$
\begin{aligned}
(H \otimes H)(|0\rangle \otimes X|0\rangle) &= H|0\rangle \otimes H|1\rangle \\
&= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
&= \frac{1}{2} \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) \\
&= \frac{1}{2} \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \otimes \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \\
&= \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)
\end{aligned}
$$

APPLY ORACLE U_F



where

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

MATHEMATICALLY

Precisly,

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

MATHEMATICALLY

If $f(0) = 0$, then

$$|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle = |0\rangle - |1\rangle.$$

If $f(0) = 1$, then

$$|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle = |1\rangle - |0\rangle = -(|0\rangle - |1\rangle).$$

Thus we can write

$$|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle = (-1)^{f(0)}(|0\rangle - |1\rangle).$$

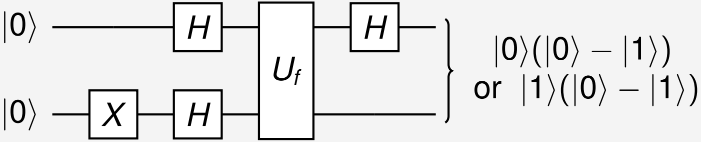
Similar reasoning applies for $f(1)$.

MATHEMATICALLY

Then we can continue

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

HADAMARD GATE TO FIRST QUBIT



MATHEMATICALLY

Let's multiply the current state by $(-1)^{f(0)}$:

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

- Multiplying the state with the constant is fine because it changes only the global phase. The relative phase stays the same.

MATHEMATICALLY

Now we have two cases. First, if f is constant, then $f(0) = f(1)$ and we obtain

$$
\begin{aligned}
&\frac{|0\rangle(|0\rangle - |1\rangle) + (-1)^{f(0)+f(1)}|1\rangle(|0\rangle - |1\rangle)}{2} \\
&= \frac{|0\rangle(|0\rangle - |1\rangle) + (-1)^{2f(0)}|1\rangle(|0\rangle - |1\rangle)}{2} \\
&= \frac{|0\rangle(|0\rangle - |1\rangle) + |1\rangle(|0\rangle - |1\rangle)}{2} \\
&= \frac{(|0\rangle + |1\rangle)}{2}(|0\rangle - |1\rangle).
\end{aligned}
$$

MATHEMATICALLY

Second, if f is balanced, then $f(0) \neq f(1)$ and we obtain $(-1)^{f(0)+f(1)} = (-1)^{(0+1)} = -1$. Thus

$$\begin{aligned}
&\frac{|0\rangle(|0\rangle - |1\rangle) + (-1)^{f(0)+f(1)}|1\rangle(|0\rangle - |1\rangle)}{2} \\
&= \frac{|0\rangle(|0\rangle - |1\rangle) - |1\rangle(|0\rangle - |1\rangle)}{2} \\
&= \frac{(|0\rangle - |1\rangle)}{2}(|0\rangle - |1\rangle).
\end{aligned}$$

APPLY LAST HADAMARD GATE

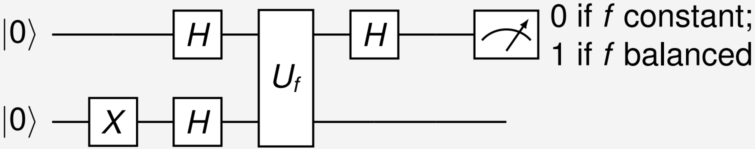
Based on the previous lessons, we know that Hadamard-gate is its own inverse. When we apply Hadamard again to the first gate, we obtain

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

and

$$H\left(\frac{|0\rangle - |1\rangle}{2}\right)(|0\rangle - |1\rangle)$$
$$= |1\rangle(|0\rangle - |1\rangle).$$

MEASUREMENT



MATHEMATICALLY

Now it is easy to read the result of the algorithm. If we measure the first qubit of the state,

$$|0\rangle(|0\rangle - |1\rangle)$$

we measure 0 with 100% probability. This happens only if f is constant. Otherwise we mesure the state

$$|1\rangle(|0\rangle - |1\rangle)$$

and always obtain 1. In this case f is balanced.

LINKS TO RUNNING EXAMPLES IN QUIRK AND QISKIT

- Example of f being identity
- Example of f being swap
- Example of f being constant 0
- Example of f constant being 1
- Running example in Qiskit

REFERENCES I

[1] D. Koch, L. Wessing, and P. M. Alsing.
Introduction to coding quantum algorithms: A tutorial series using qiskit.
arXiv:1903.04359 [quant-ph], Mar 2019.
arXiv: 1903.04359.

[2] C. Lectures.
A practical introduction to quantum computing - Elias Fernandez-Combarro Alvarez - (3/7).
Feb 2020.