# LESSON 5.2

## DEUTSCH-JOZSA & BERNSTEIN-VAZIRANI ALGORITHMS

Valter Uotila

# OUTLINE

Introduction

Generalized oracle

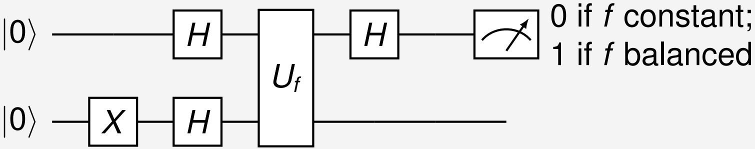
Deutsch-Jozsa algorithm

Bernstein-Vazirani algorithm

Demonstrations

# RECAP FROM PREVIOUS LESSON

We learned that the simplest version of Deutsch's algorithm can be implemented with the following circuit




We use bit strings and their corresponding 10-base numbers interchangeably, e.g., the bit string 1101 is 13 in 10-base.

# MOTIVATION & BACKGROUND

- In 1992 Deutsch and Jozsa [4] developed generalization of Deutsch's algorithm
- Deutsch-Jozsa's algorithm [1] applies for any function $f\colon \{0,1\}^n \rightarrow \{0,1\}$ where $n > 0$ which is either constant or maps half of the values to 1 and half of the values to 0.
- In order that the algorithm works, it is important that the ratio is half-half which we will see later

# EXAMPLE

**Example**

The function $f\colon \{00, 01, 10, 11\} \to \{1, 0\}$ mapping all the domain bit strings to 1 is one example of a function that Deutsch-Jozsa algorithm is able to detect. We need just one oracle call to say that it is constant.

# MOTIVATION & BACKGROUND

**Deutsch-Jozsa's problem**

Determine if $f\colon \{0,1\}^n \to \{0,1\}$ for $n \in \mathbb{N}$ is constant or balanced function.

- To get classically certainly correct result, we need to evaluate the function at every point
- As in the Deutsch's algorithm, quantumly we can solve the problem with just one oracle call

# DEUTSCH VS. DEUTSCH-JOZSA

## Deutsch's algorithm – 2 qubits

1. Prepare state $|0\rangle \otimes |1\rangle$
2. Hadamard transform for all qubits
3. Apply oracle of size 4
4. Apply Hadamard gate to the first qubit
5. Measure the first qubit

# DEUTSCH VS. DEUTSCH-JOZSA

## Deutsch-Jozsa's algorithm – $n + 1$ qubits

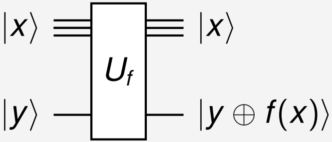
1. Prepare state $|\underbrace{0\ldots0}_{n \text{ times}}\rangle \otimes |1\rangle$
2. Hadamard transform for all qubits
3. Apply oracle of size $2^{n+1}$
4. Apply Hadamard transform to $n$ first qubits
5. Measure the $n$ first qubits

# ORACLE

Again we define a unitary operator (oracle) $U_f$ so that

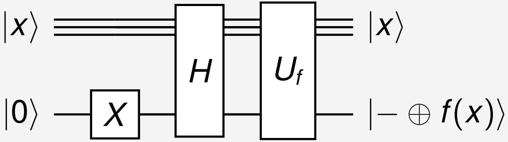
$$U_f|xy\rangle = |x\rangle|y \oplus f(x)\rangle.$$

We see that this is exactly same as in the Deutsch's algorithm. Drawing the circuit:



# ORACLE

Let's study how we should design oracle so that it would implement operator $U_f$ correctly. Recall that we are in the following state at the moment of applying the oracle:




So, we are interested in understanding $|- \oplus f(x)\rangle$.

# ORACLE'S ACTION

Recall that as in Deutsch's algorithm, we prepare the last qubit in the state $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ before we apply the oracle. We can calculate that $X|-\rangle = -|-\rangle$.

---

Let $x$ be a bit string and $f(x) = 0$.

$$\begin{aligned} U_f|x\rangle|-\rangle &= |x\rangle|-\oplus f(x)\rangle \\ &= |x\rangle|-\oplus 0\rangle \\ &= |x\rangle|-\rangle \end{aligned}$$

Let $x$ be a bit string and $f(x) = 1$.

$$\begin{aligned} U_f|x\rangle|-\rangle &= |x\rangle|-\oplus f(x)\rangle \\ &= |x\rangle|-\oplus 1\rangle \\ &= |x\rangle X|-\rangle \\ &= -|x\rangle|-\rangle \end{aligned}$$

# MEANING OF THE PREVIOUS CALCULATIONS

- Previous calculations showed that by applying the oracle, we introduced a flip to the phase always when $f(x) = 1$.

# MEANING OF THE PREVIOUS CALCULATIONS

- Previous calculations showed that by applying the oracle, we introduced a flip to the phase always when $f(x) = 1$.
- Allegory: we have a lock and a set of possible keys. Some of the keys work, and some don't. Now we can mark the working keys with $-1$ and others with 1 [2]

# MEANING OF THE PREVIOUS CALCULATIONS

- Previous calculations showed that by applying the oracle, we introduced a flip to the phase always when $f(x) = 1$.
- Allegory: we have a lock and a set of possible keys. Some of the keys work, and some don't. Now we can mark the working keys with $-1$ and others with 1 [2]
- Similarly, when $f$ maps a bit string to 1, we mark this with X-gate in the matrix representation of the oracle. This change introduced flipped phase for the corresponding state.

# ORACLE EXAMPLE

Let $f\colon \{00, 01, 10, 11\} \to \{1, 0\}$ be the function which maps the first two elements to 1 and the last two elements to 0. The matrix

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

# COMPARE TO DEUTSCH'S ALGORITHM

- When we created oracles for Deutsch algorithm, we just noticed that they behave certain way and encoded this behaviour
- When we have more qubits, this becomes harder and we need some algorithm to construct oracles

# COMPARE TO DEUTSCH: CASE 1

In the case that $f$ is identity: $0 \mapsto 0$ and $1 \mapsto 1$

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

# COMPARE TO DEUTSCH: CASE 2

In the case that $f$ is swap: $0 \mapsto 1$ and $1 \mapsto 0$

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

# COMPARE TO DEUTSCH: CASE 3

In the case that $f$ is constant 0: $0 \mapsto 0$ and $1 \mapsto 0$

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

# COMPARE TO DEUTSCH: CASE 4

In the case that $f$ is constant 1: $0 \mapsto 1$ and $1 \mapsto 1$

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

# ORACLE

- If we do not care to use the basic gates, we can use the matrix representation to implement any oracle easily. Qiskit implements a feature that creates a gate based on a given unitary matrix. I use that in the Qiskit demonstration.

# ORACLE

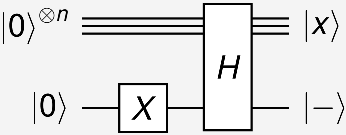
- If we do not care to use the basic gates, we can use the matrix representation to implement any oracle easily. Qiskit implements a feature that creates a gate based on a given unitary matrix. I use that in the Qiskit demonstration.
- On the other hand, we do not necessarily have such functionality. Then we can use X-gates and multi-control-CNOT-gates. See the Quirk demonstrations.

# ORACLE

- If we do not care to use the basic gates, we can use the matrix representation to implement any oracle easily. Qiskit implements a feature that creates a gate based on a given unitary matrix. I use that in the Qiskit demonstration.
- On the other hand, we do not necessarily have such functionality. Then we can use X-gates and multi-control-CNOT-gates. See the Quirk demonstrations.
- In some cases, multi-control-CNOT-gates are not available. Then we can implement the equivalent circuit using Toffoli and CNOT-gates.

# PREPARE ANCILLA QUBIT AND APPLY HADAMARD TRANSFORM

The following circuit prepares the ancilla qubit in the state $|-\rangle$ and applies Hadamard transform:




$$|x\rangle|-\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle|-\rangle$$

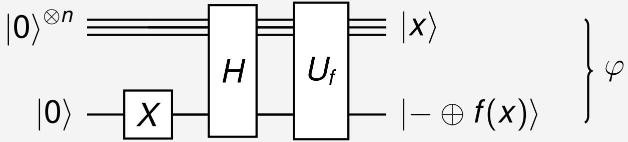
# MATHEMATICALLY

We start at state $|0\rangle^{\otimes n}|0\rangle$. After applying X-gate to the ancilla qubit we obtain $|0\rangle^{\otimes n}|1\rangle$. Then we apply Hadamard transform:

$$\begin{aligned}
H^{\otimes n+1}(|0\rangle^{\otimes n}|1\rangle) &= H^{\otimes n}|0\rangle^{\otimes n}H|1\rangle) \\
&= \frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}} \\
&= \frac{1}{\sqrt{2^{n+1}}}\sum_{i=0}^{2^n-1}|i\rangle(|0\rangle-|1\rangle).
\end{aligned}$$

# APPLY ORACLE

Next we apply the oracle $U_f$:




where $\varphi = \frac{1}{\sqrt{2^{n+1}}} \sum_{i=0}^{2^n-1} (-1)^{f(i)} |i\rangle(|0\rangle - |1\rangle)$.

# MATHEMATICALLY

Recalling that $U_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle$, we obtain

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

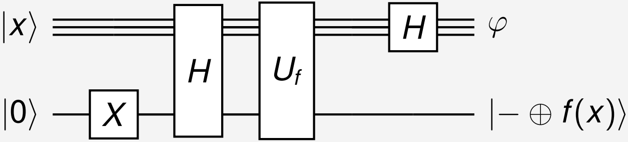
# MATHEMATICALLY

Now $f(i) = 0$ or $f(i) = 1$. Following similar reasoning as in the case of Deutsch's algorithm, we obtain

$$U_f(|x\rangle|-\rangle) = \frac{1}{\sqrt{2^{n+1}}}\sum_{i=0}^{2^n-1}(-1)^{f(i)}|i\rangle(|0\rangle - |1\rangle).$$

# HADAMARD TRANSFORM AGAIN

Next, we apply Hadamard transform for the first $n$ qubits.
We can ignore the $|-\rangle$ part of the state.




where $\varphi = \frac{1}{2^n} \sum_{i=0}^{2^n-1} \left( \sum_{j=0}^{2^n-1} (-1)^{f(i)} (-1)^{i \cdot j} \right) |j\rangle$.

# MATHEMATICALLY

$$H^{\otimes n}\left(\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}(-1)^{f(i)}|i\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}(-1)^{f(i)}H^{\otimes n}|i\rangle$$

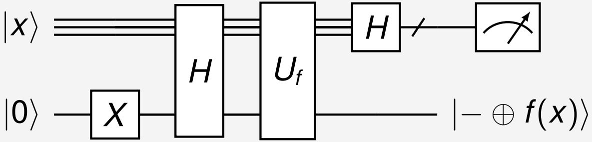
$$= \frac{1}{2^n}\sum_{i=0}^{2^n-1}(-1)^{f(i)}\left(\sum_{j=0}^{2^n-1}(-1)^{i\cdot j}|j\rangle\right)$$

$$= \frac{1}{2^n}\sum_{i=0}^{2^n-1}\left(\sum_{j=0}^{2^n-1}(-1)^{f(i)}(-1)^{i\cdot j}\right)|j\rangle,$$

where $i \cdot j = i_0 j_0 \otimes \ldots \otimes i_{n-1} j_{n-1}$ is bitwise product.

# MEASUREMENT

Finally we measure the first $n$ qubits:



# MATHEMATICALLY

We see that if $f$ is constant, then we measure $\underbrace{0\cdots0}_{n \text{ times}}$ with probability 1. This is true because when $j = 0$, the amplitude of $|0\rangle$ is

$$\left|\frac{1}{2^n}\sum_{i=0}^{2^n-1}(-1)^{f(i)}\right|.$$

# MATHEMATICALLY

If $f(i) = 0$, then

$$\left| \frac{1}{2^n} \sum_{i=0}^{2^n-1} 1 \right| = \frac{2^n}{2^n} = 1.$$

If $f(i) = 1$, then

$$\left| \frac{1}{2^n} \sum_{i=0}^{2^n-1} -1 \right| = \left| -\frac{2^n}{2^n} \right| = 1.$$

# MATHEMATICALLY

If $f$ is balanced, then we measure $\underbrace{0 \cdots 0}_{n \text{ times}}$ with probability 0.

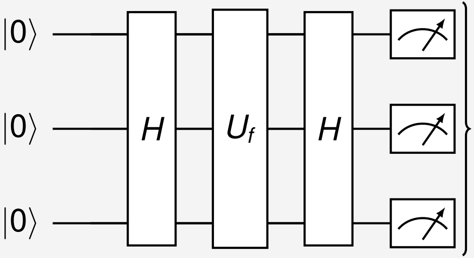
In order that the algorithm produces 100% correct solution half of the values of $f$ needs to be 0s and another half 1s. The amplitude of $|0\rangle$ is

$$\left| \frac{1}{2^n} \sum_{i=0}^{2^n-1} (-1)^{f(i)} \right| .$$

We see that $(-1)^{f(i)}$ terms cancel each other because half of them evaluate to 1 (when $f(i) = 0$) and another half to $-1$ (when $f(i) = 1$). Thus the amplitude is 0.

# DO WE NEED THE ANCILLA QUBIT?

We can use the following kind of circuit [2] to implement the Deutsch-Jozsa algorithm without ancilla qubit:




if all measured as 0, then $f$ is constant; otherwise balanced

# DO WE NEED THE ANCILLA QUBIT?

The oracle $U_f$ is implemented so that for every element $x_{i,i}$ in its diagonal, we have $x_{i,i} = -1$ if $f(i) = 1$ and $x_{i,i} = 1$, if $f(i) = 0$. It would be interesting to discuss pros and cons of each implementation. Besides, the third possible circuit to implement Deutsch-Jozsa algorithm with ancilla qubit is represented in [5].

# MOTIVATION & BACKGROUND

- Bernstein-Vazirani algorithm [3] was represented in 1992.
- It is a restricted version of Deutsch-Jozsa algorithm

# BERNSTEIN-VAZIRANI PROBLEM

In the Deutsch-Jozsa problem, the function $f$ was relatively general, but in the Bernstein-Vazirani problem, we restrict it more:

**Problem**

Let $f\colon \{0,1\}^n \to \{0,1\}$ for $n \in \mathbb{N}$ be a function defined by

$$f_y(x) = x \cdot y \mod 2,$$

where $x \cdot y$ is the bitwise dot product. What is the value of $y$?

# EXAMPLE

For example, $x = 1011$ and $y = 1001$, then

$$\begin{aligned} f_y(x) &= x \cdot y \mod 2 \\ &= (1 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 1 \cdot 1) \mod 2 \\ &= 2 \mod 2 = 0. \end{aligned}$$

# BERNSTEIN-VAZIRANI ALGORITHM

- The algorithm is exactly the same
- The oracle is exactly the same $U_f|xy\rangle = |x\rangle|y \oplus f(x)\rangle$
- The demo shows more details how the algorithm works

# LINKS TO RUNNING EXAMPLES IN QUIRK AND QISKIT

- Quirk Deutsch-Jozsa algorithm example: constant 1 function $f$
- Quirk Deutsch-Jozsa algorithm example: balanced function $f$
- Qiskit implementation of Deutsch-Jozsa algorithm
- Pennylane implementation of Deutsch-Jozsa algorithm without ancilla qubit
- Qiskit implementation of Bernstein-Vazirani algorithm

# REFERENCES I


[1] Qiskit textbook - deutsch-jozsa algorithm, 2022.

[2] Xanadu quantum codebook - learn quantum computing interactively online with pennylane, 2022.

[3] E. Bernstein and U. Vazirani.
Quantum complexity theory.
*SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[4] D. Deutsch and R. Jozsa.
Rapid solution of problems by quantum computation.
*Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, Dec 1992.

[5] C. Lectures.
*A practical introduction to quantum computing - Elias Fernandez-Combarro Alvarez - (4/7)*.
Feb 2020.