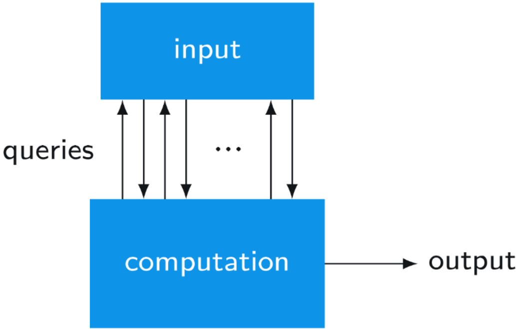
Quantum Computing Seminar 6

YongHyun "Aeren" An

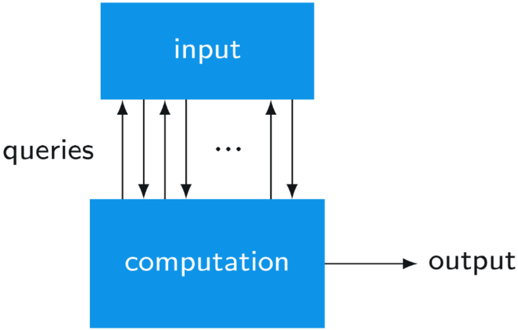
Samsung Research

December 16, 2024
December 23, 2024

Query Model of Computation

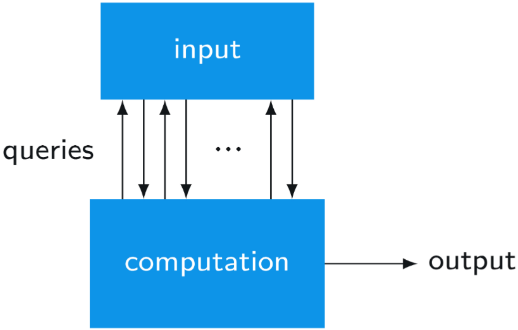


Query Model of Computation



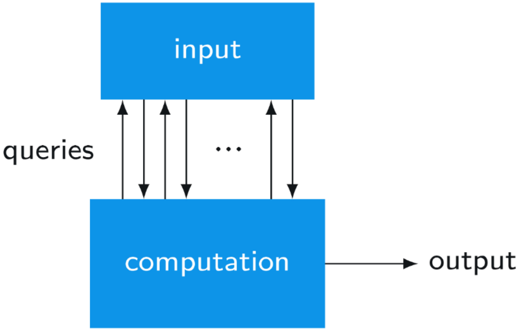
- In query model of computation, we access the input by making **queries**.

Query Model of Computation



- In query model of computation, we access the input by making **queries**.
- We refer to the input as being provided by an **oracle** or a **blackbox**.

Query Model of Computation



- In query model of computation, we access the input by making **queries**.
- We refer to the input as being provided by an **oracle** or a **blackbox**.
- The oracle is represented as a function $f : \{0, 1\}^n \to \{0, 1\}^m$, for some fixed integer n and m.

Query Model of Computation

- Q) Is $U_f : |s\rangle \mapsto |f(s)\rangle$ a valid gate in a quantum circuit?

Query Model of Computation

- Q) Is $U_f : |s\rangle \mapsto |f(s)\rangle$ a valid gate in a quantum circuit?
- Q) Is $U_f : |t, s\rangle \mapsto |t \oplus f(s), s\rangle$ a valid gate in a quantum circuit?

Query Model of Computation

Accessing Oracle

Query Model of Computation

Accessing Oracle

- In **query model of classical computation**, the oracle is accessed through calling the function $C_f(s) := f(s)$.

Query Model of Computation

Accessing Oracle

- In **query model of classical computation**, the oracle is accessed through calling the function $C_f(s) := f(s)$.
- In **query model of quantum computation**, the oracle is accessed through the gate $U_f(|t, s\rangle) := |t \oplus f(s), s\rangle$

Query Model of Computation

Accessing Oracle

- In **query model of classical computation**, the oracle is accessed through calling the function $C_f(s) := f(s)$.
- In **query model of quantum computation**, the oracle is accessed through the gate $U_f(|t, s\rangle) := |t \oplus f(s), s\rangle$
- The **cost of a query model classical algorithm** is the number of C_f called.

Query Model of Computation

Accessing Oracle

- In **query model of classical computation**, the oracle is accessed through calling the function $C_f(s) := f(s)$.
- In **query model of quantum computation**, the oracle is accessed through the gate $U_f(|t, s\rangle) := |t \oplus f(s), s\rangle$
- The **cost of a query model classical algorithm** is the number of C_f called.
- The **cost of a query model quantum algorithm** is the number of U_f gate used.

Query Model of Computation

Accessing Oracle

- In **query model of classical computation**, the oracle is accessed through calling the function $C_f(s) := f(s)$.
- In **query model of quantum computation**, the oracle is accessed through the gate $U_f(|t, s\rangle) := |t \oplus f(s), s\rangle$
- The **cost of a query model classical algorithm** is the number of C_f called.
- The **cost of a query model quantum algorithm** is the number of U_f gate used.
- We're going to see examples of query model quantum algorithms that outperform query model classical algorithms.

Query Model of Computation

Why care about the query model?

Query Model of Computation

Why care about the query model?

1. Query model algorithms can rule out fast quantum algorithms.

Query Model of Computation

Why care about the query model?

1. Query model algorithms can rule out fast quantum algorithms.
2. The query model of classical computing is well-studied.

Query Model of Computation

Why care about the query model?

1. Query model algorithms can rule out fast quantum algorithms.
2. The query model of classical computing is well-studied.
3. It gives insight into how quantum algorithms work. Instantiating the "black box" in terms of quantum gates can lead to fast quantum algorithms.

Phase kickback

Phase kickback

- For all $a, b \in \{ 0, 1 \}$, it's easy to verify that $|a \oplus b\rangle = X^b |a\rangle$.

Query Model of Computation

Phase kickback

- For all $a, b \in \{0, 1\}$, it's easy to verify that $|a \oplus b\rangle = X^b |a\rangle$.
- We can now see that for all $a, b \in \{0, 1\}$,

$$U_f(|b, a\rangle) = |b \oplus f(a)\rangle \otimes |a\rangle = X^{f(a)} |b, a\rangle$$

Query Model of Computation

Phase kickback

- For all $a, b \in \{ 0, 1 \}$, it's easy to verify that $|a \oplus b\rangle = X^b |a\rangle$.
- We can now see that for all $a, b \in \{ 0, 1 \}$,

$$U_f (|b, a\rangle) = |b \oplus f(a)\rangle \otimes |a\rangle = X^{f(a)} |b, a\rangle$$

- Since it holds for all $b \in \{ 0, 1 \}$, it must hold for all 1-qubit state $|u\rangle$

$$U_f (|u, a\rangle) = X^{f(a)} |u, a\rangle$$

Query Model of Computation

Phase kickback

- For all $a, b \in \{0, 1\}$, it's easy to verify that $|a \oplus b\rangle = X^b |a\rangle$.
- We can now see that for all $a, b \in \{0, 1\}$,

$$U_f(|b, a\rangle) = |b \oplus f(a)\rangle \otimes |a\rangle = X^{f(a)} |b, a\rangle$$

- Since it holds for all $b \in \{0, 1\}$, it must hold for all 1-qubit state $|u\rangle$

$$U_f(|u, a\rangle) = X^{f(a)} |u, a\rangle$$

- Therefore,

$$U_f(|-, a\rangle) = X^{f(a)} |-\rangle \otimes |a\rangle = (-1)^{f(a)} |-, a\rangle$$

Deutsch's Algorithm

Deutsch's Algorithm

Definition (constant and balanced function)

A function $f : \{ 0, 1 \}^n \to \{ 0, 1 \}$ is

- **constant** if $f(s) = f(t)$ for all $s, t \in \{ 0, 1 \}^n$, and
- **balanced** if $|\{ s : f(s) = 0 \}| = |\{ t : f(t) = 1 \}|$.

Deutsch's Algorithm

Definition (constant and balanced function)

A function $f : \{0,1\}^n \to \{0,1\}$ is

- **constant** if $f(s) = f(t)$ for all $s, t \in \{0,1\}^n$, and
- **balanced** if $|\{ s : f(s) = 0 \}| = |\{ t : f(t) = 1 \}|$.

Deutsch's problem

Input	a function $f : \{0,1\} \to \{0,1\}$
Output	0 if f is constant, 1 if f is balanced

Deutsch's Algorithm

Classical algorithm

Deutsch's Algorithm

Classical algorithm

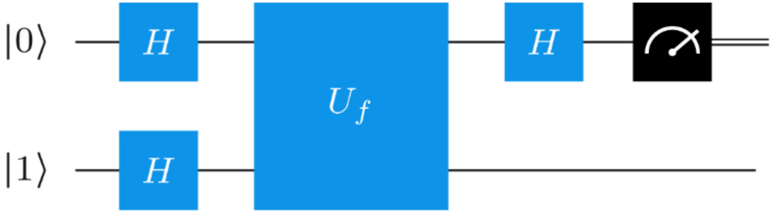
- Any classical algorithm must make at least 2 oracle calls, because regardless of querying $f(0)$ or $f(1)$, it must know the other value to determine the answer.

Deutsch's Algorithm

Quantum algorithm (Deutsch's algorithm)

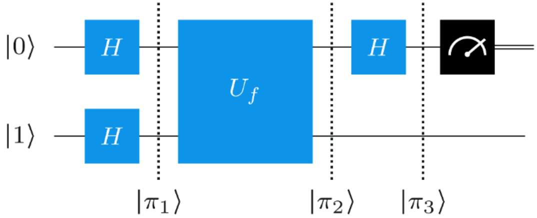
Deutsch's Algorithm

Quantum algorithm (Deutsch's algorithm)

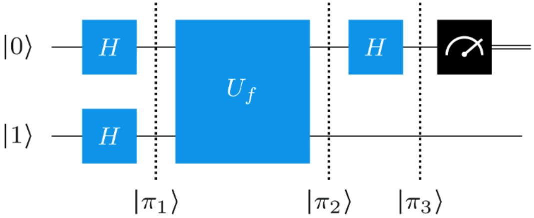


$$\begin{cases} 0 & \text{if } f \text{ is constant} \\ 1 & \text{if } f \text{ is balanced} \end{cases}$$

Quantum algorithm (Deutsch's algorithm)

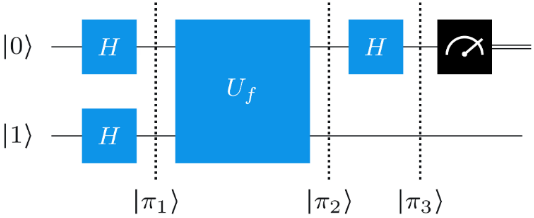


Quantum algorithm (Deutsch's algorithm)



- $|\pi_1\rangle = (H \otimes H)\,|10\rangle = |-,+\rangle = \frac{1}{\sqrt{2}}\,|-,0\rangle + \frac{1}{\sqrt{2}}\,|-,1\rangle$

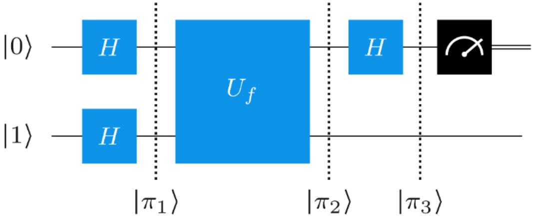
Quantum algorithm (Deutsch's algorithm)



- $|\pi_1\rangle = (H \otimes H)\,|10\rangle = |-,+\rangle = \frac{1}{\sqrt{2}}\,|-,0\rangle + \frac{1}{\sqrt{2}}\,|-,1\rangle$
- $|\pi_2\rangle = U_f\,|\pi_1\rangle = \frac{1}{\sqrt{2}} U_f\,|-,0\rangle + \frac{1}{\sqrt{2}} U_f\,|-,1\rangle = \frac{1}{\sqrt{2}}(-1)^{f(0)}\,|-,0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}\,|-,1\rangle$

Deutsch's Algorithm

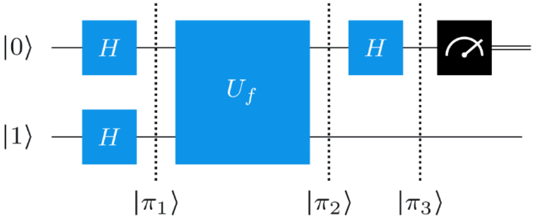
Quantum algorithm (Deutsch's algorithm)



- $|\pi_1\rangle = (H \otimes H)|10\rangle = |-,+\rangle = \frac{1}{\sqrt{2}}|-,0\rangle + \frac{1}{\sqrt{2}}|-,1\rangle$
- $|\pi_2\rangle = U_f|\pi_1\rangle = \frac{1}{\sqrt{2}}U_f|-,0\rangle + \frac{1}{\sqrt{2}}U_f|-,1\rangle = \frac{1}{\sqrt{2}}(-1)^{f(0)}|-,0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}|-,1\rangle$

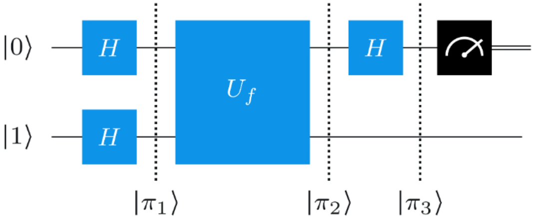
  $= (-1)^{f(0)}|-\rangle \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\right) = (-1)^{f(0)}|-\rangle \otimes \begin{cases} |+\rangle & \text{if } f(0) = f(1) \\ |-\rangle & \text{otherwise} \end{cases}$

Quantum algorithm (Deutsch's algorithm)



- $|\pi_2\rangle = (-1)^{f(0)}\,|-\rangle \otimes \begin{cases} |+\rangle & \text{if } f(0) = f(1) \\ |-\rangle & \text{otherwise} \end{cases}$

Quantum algorithm (Deutsch's algorithm)



- $|\pi_2\rangle = (-1)^{f(0)} |-\rangle \otimes \begin{cases} |+\rangle & \text{if } f(0) = f(1) \\ |-\rangle & \text{otherwise} \end{cases}$
- $|\pi_3\rangle = \begin{cases} (-1)^{f(0)} |-\rangle \otimes |0\rangle & \text{if } f(0) = f(1) \\ (-1)^{f(0)} |-\rangle \otimes |1\rangle & \text{otherwise} \end{cases}$

Summary of results for the Deutsch's problem

Model	Classical (Deterministic)	Classical (Probabilistic)	Quantum
Cost	2	2	1

Deutsch-Jozsa Algorithm

Deutsch-Jozsa Algorithm

Definition (addition and inner product of bitstrings)

For $x = x_{n-1} \cdots x_0$ and $y = y_{n-1} \cdots y_0 \in \{\, 0, 1 \,\}^n$,

- $x \oplus y = (x_{n-1} \oplus y_{n-1}) \cdots (x_0 \oplus y_0)$
- $x \cdot y = x_{n-1} \cdot y_{n-1} \oplus \cdots \oplus x_0 \cdot y_0$

Deutsch-Jozsa Algorithm

Definition (addition and inner product of bitstrings)

For $x = x_{n-1} \cdots x_0$ and $y = y_{n-1} \cdots y_0 \in \{0,1\}^n$,

- $x \oplus y = (x_{n-1} \oplus y_{n-1}) \cdots (x_0 \oplus y_0)$
- $x \cdot y = x_{n-1} \cdot y_{n-1} \oplus \cdots \oplus x_0 \cdot y_0$

- For $a \in \{0,1\}$,

$$H|a\rangle = \frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} (-1)^{a \cdot b} |b\rangle$$

Deutsch-Jozsa Algorithm

Definition (addition and inner product of bitstrings)

For $x = x_{n-1} \cdots x_0$ and $y = y_{n-1} \cdots y_0 \in \{0, 1\}^n$,

- $x \oplus y = (x_{n-1} \oplus y_{n-1}) \cdots (x_0 \oplus y_0)$
- $x \cdot y = x_{n-1} \cdot y_{n-1} \oplus \cdots \oplus x_0 \cdot y_0$

- For $a \in \{0, 1\}$,

$$H |a\rangle = \frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} (-1)^{a \cdot b} |b\rangle$$

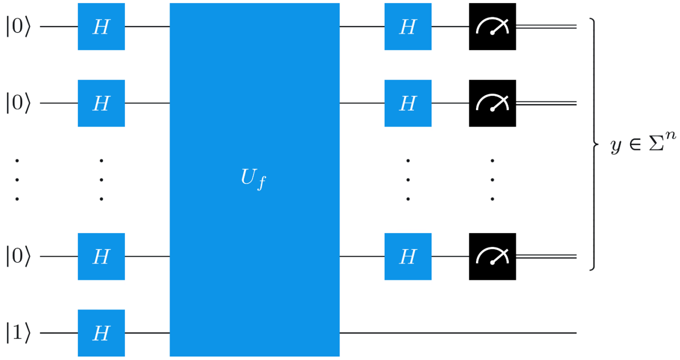
- For $x \in \{0, 1\}^n$,

$$\begin{aligned} H^{\otimes n} |x\rangle &= H |x_{n-1}\rangle \otimes \cdots \otimes H |x_0\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \end{aligned}$$

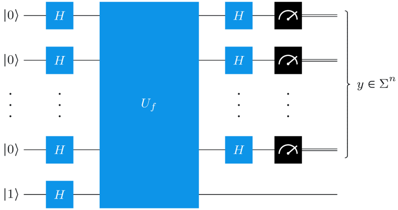
Deutsch-Jozsa Algorithm

Deutsch-Jozsa Algorithm

Deutsch-Jozsa Algorithm

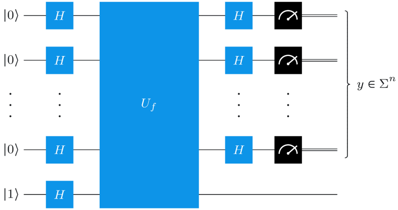


Deutsch-Jozsa Algorithm



1. Initial state: $|10\cdots0\rangle$

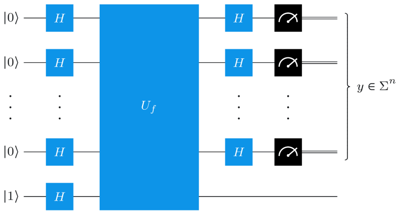
Deutsch-Jozsa Algorithm



Deutsch-Jozsa Algorithm

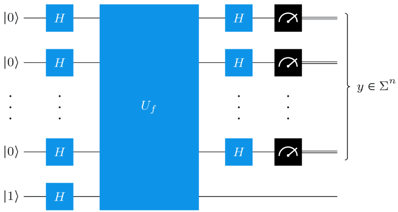
1. Initial state: $|10\cdots0\rangle$
2. After the 1st layer: $|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$

Deutsch-Jozsa Algorithm



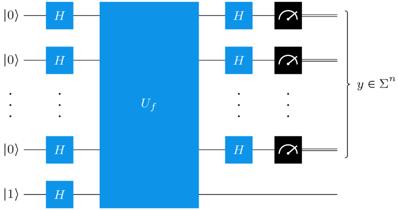
Deutsch-Jozsa Algorithm

Deutsch-Jozsa Algorithm



1. Initial state: $|10\cdots0\rangle$
2. After the 1st layer: $|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$
3. After the 2nd layer: $|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle$

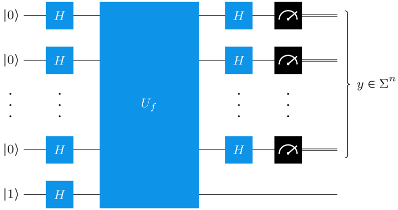
Deutsch-Jozsa Algorithm



1. Initial state: $|10\cdots0\rangle$
2. After the 1st layer: $|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$
3. After the 2nd layer: $|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle$
4. After the 3rd layer:

$$|-\rangle \otimes \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}^n} (-1)^{f(x)+x\cdot y} |y\rangle$$

Deutsch-Jozsa Algorithm



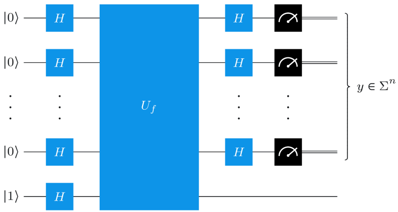
1. Initial state: $|10\cdots0\rangle$
2. After the 1st layer: $|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$
3. After the 2nd layer: $|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle$
4. After the 3rd layer:

$$|-\rangle \otimes \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}^n} (-1)^{f(x)+x\cdot y} |y\rangle$$

5. Probability of the result being $r \in \{0,1\}^n$:

Deutsch-Jozsa Algorithm

Deutsch-Jozsa Algorithm



1. Initial state: $|10\cdots0\rangle$
2. After the 1st layer: $|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$
3. After the 2nd layer: $|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle$
4. After the 3rd layer:

$$|-\rangle \otimes \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}^n} (-1)^{f(x)+x\cdot y} |y\rangle$$

5. Probability of the result being $r \in \{0,1\}^n$:

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)+x\cdot r} \right|^2$$

Deutsch-Jozsa Algorithm

We'll look at two problems solvable by the Deutsch-Jozsa algorithm.

Deutsch-Jozsa Algorithm

We'll look at two problems solvable by the Deutsch-Jozsa algorithm.

Deutsch-Jozsa problem

Input	a function $f : \{0,1\}^n \to \{0,1\}$ which is either constant or balanced
Output	0 if f is constant, 1 if f is balanced

Deutsch-Jozsa Algorithm

Classical algorithm (deterministic)

Deutsch-Jozsa Algorithm

Classical algorithm (deterministic)

- Any classical algorithm must make at least $2^{n-1}+1$ oracle calls, because regardless of the result of first 2^{n-1} queries, the answer can still be either constant or balanced.

Deutsch-Jozsa Algorithm

Classical algorithm (deterministic)

- Any classical algorithm must make at least $2^{n-1}+1$ oracle calls, because regardless of the result of first 2^{n-1} queries, the answer can still be either constant or balanced.

Classical algorithm (probabilistic)

Deutsch-Jozsa Algorithm

Classical algorithm (deterministic)

- Any classical algorithm must make at least $2^{n-1} + 1$ oracle calls, because regardless of the result of first 2^{n-1} queries, the answer can still be either constant or balanced.

Classical algorithm (probabilistic)

- We can randomly choose a bitstring and query $k = 30$ times.

Deutsch-Jozsa Algorithm

Classical algorithm (deterministic)

- Any classical algorithm must make at least $2^{n-1}+1$ oracle calls, because regardless of the result of first 2^{n-1} queries, the answer can still be either constant or balanced.

Classical algorithm (probabilistic)

- We can randomly choose a bitstring and query $k = 30$ times.
- If f is constant, all k outputs will be the same.

Deutsch-Jozsa Algorithm

Classical algorithm (deterministic)

- Any classical algorithm must make at least $2^{n-1} + 1$ oracle calls, because regardless of the result of first 2^{n-1} queries, the answer can still be either constant or balanced.

Classical algorithm (probabilistic)

- We can randomly choose a bitstring and query $k = 30$ times.
- If f is constant, all k outputs will be the same.
- If f is balanced, the outputs will contain both 0 and 1 with probability $1 - \frac{1}{2^{29}}$.

Deutsch-Jozsa Algorithm

Classical algorithm (deterministic)

- Any classical algorithm must make at least $2^{n-1} + 1$ oracle calls, because regardless of the result of first 2^{n-1} queries, the answer can still be either constant or balanced.

Classical algorithm (probabilistic)

- We can randomly choose a bitstring and query $k = 30$ times.
- If f is constant, all k outputs will be the same.
- If f is balanced, the outputs will contain both 0 and 1 with probability $1 - \frac{1}{2^{29}}$.
- Therefore, judging that f is constant or not depending on whether the output contains both 0 or 1 has failure probability equal or less than $\frac{1}{2^{29}}$.

Deutsch-Jozsa Algorithm

Quantum algorithm (Deutsch-Jozsa Algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) + x \cdot r} \right|^2$$

Quantum algorithm (Deutsch-Jozsa Algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) + x \cdot r} \right|^2$$

We focus on the probability for $r = 0 \cdots 0$.

$$\mathcal{P}(0 \cdots 0) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} \right|^2 = \begin{cases} 0 & \text{if } f \text{ is balanced} \\ 1 & \text{if } f \text{ is constant} \end{cases}$$

Quantum algorithm (Deutsch-Jozsa Algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) + x \cdot r} \right|^2$$

We focus on the probability for $r = 0 \cdots 0$.

$$\mathcal{P}(0 \cdots 0) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} \right|^2 = \begin{cases} 0 & \text{if } f \text{ is balanced} \\ 1 & \text{if } f \text{ is constant} \end{cases}$$

Therefore, we judge that f is constant if and only if the output is $0 \cdots 0$.

Summary of results for the Deutsch-Jozsa problem

Model	Classical (Deterministic)	Classical (Probabilistic)	Quantum
Cost	$2^{n-1} + 1$	Some constant	1

Deutsch-Jozsa Algorithm

Bernstein-Vazirani problem

Input	a function $f : \{0, 1\}^n \to \{0, 1\}$ satisfying $f(x) = s \cdot x$ for some fixed bitstring s
Output	bitstring s

Deutsch-Jozsa Algorithm

Classical algorithm

Deutsch-Jozsa Algorithm

Classical algorithm

- Any classical algorithm must make at least n oracle calls, because it needs to distinguish 2^n possible cases.

Deutsch-Jozsa Algorithm

Classical algorithm

- Any classical algorithm must make at least n oracle calls, because it needs to distinguish 2^n possible cases.
- On the other hand, querying all bitstrings with exactly one 1 allows us to extract s one by one.

Deutsch-Jozsa Algorithm

Classical algorithm

- Any classical algorithm must make at least n oracle calls, because it needs to distinguish 2^n possible cases.
- On the other hand, querying all bitstrings with exactly one 1 allows us to extract s one by one.
- Therefore, n oracle calls is the best we can do.

Deutsch-Jozsa Algorithm

Quantum algorithm (Deutsch-Jozsa Algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) + x \cdot r} \right|^2$$

Deutsch-Jozsa Algorithm

Quantum algorithm (Deutsch-Jozsa Algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) + x \cdot r} \right|^2$$

Since $f(x) = s \cdot x$ for some $s \in \{0, 1\}^n$,

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{(s \oplus r) \cdot x} \right|^2 = \begin{cases} 1 & \text{if } s = r \\ 0 & \text{if } s \neq r \end{cases}$$

Deutsch-Jozsa Algorithm

Quantum algorithm (Deutsch-Jozsa Algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) + x \cdot r} \right|^2$$

Since $f(x) = s \cdot x$ for some $s \in \{0, 1\}^n$,

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{(s \oplus r) \cdot x} \right|^2 = \begin{cases} 1 & \text{if } s = r \\ 0 & \text{if } s \neq r \end{cases}$$

Therefore, the measurement result is always s.

Summary of results for the Bernstein-Vazirani problem

Model	Classical (Deterministic)	Classical (Probabilistic)	Quantum
Cost	n	n	1

Simon's Algorithm

Simon's problem

Input	a function $f : \{0,1\}^n \to \{0,1\}^m$ satisfying $[f(x) = f(y)] \iff [(x = y) \vee (x \oplus s = y)]$ for all bitstring x and y, for some fixed bitstring s
Output	bitstring s

Simon's Algorithm

Classical algorithm

Classical algorithm

- If a classical algorithm had queried two distinct bitstrings x and y with $f(x) = f(y)$, it can determine $s = x \oplus y$.

Simon's Algorithm

Classical algorithm

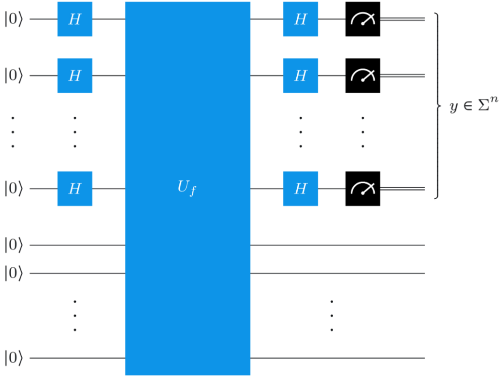
- If a classical algorithm had queried two distinct bitstrings x and y with $f(x) = f(y)$, it can determine $s = x \oplus y$.
- On the other hand, if it had queried no such pair of bitstrings, s can be any bitstring.

Simon's Algorithm

Classical algorithm

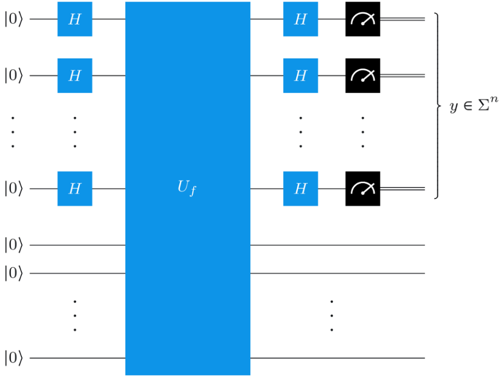
- If a classical algorithm had queried two distinct bitstrings x and y with $f(x) = f(y)$, it can determine $s = x \oplus y$.
- On the other hand, if it had queried no such pair of bitstrings, s can be any bitstring.
- By the birthday paradox, we're expected to require $\Omega(\sqrt{2^n})$ queries before finding such pair of x and y.

Quantum algorithm (Simon's algorithm)



Simon's Algorithm

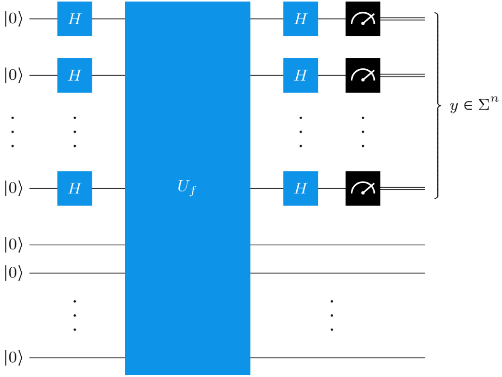
Quantum algorithm (Simon's algorithm)



- Initial state: $|0 \cdots 00 \cdots 0\rangle$

Simon's Algorithm

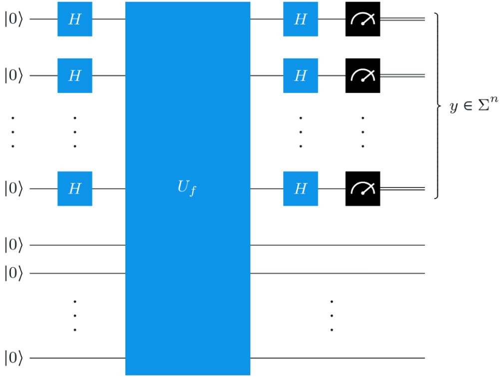
Quantum algorithm (Simon's algorithm)



- Initial state: $|0 \cdots 00 \cdots 0\rangle$
- After the 1st layer: $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |0 \cdots 0\rangle |x\rangle$

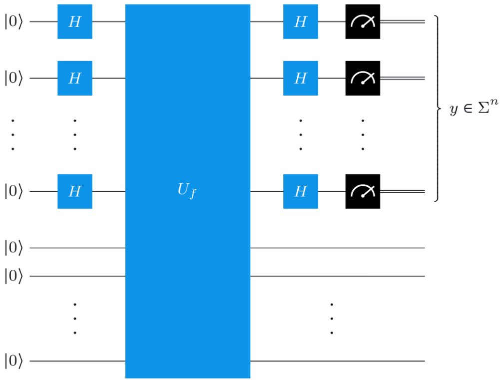
Simon's Algorithm

Quantum algorithm (Simon's algorithm)



- Initial state: $|0 \cdots 00 \cdots 0\rangle$
- After the 1st layer: $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |0 \cdots 0\rangle |x\rangle$
- After the 2nd layer: $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |f(x)\rangle |x\rangle$

Quantum algorithm (Simon's algorithm)

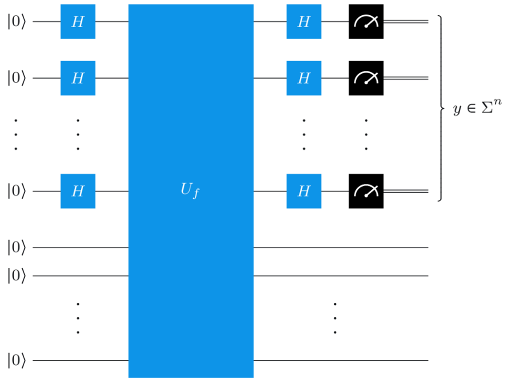


- Initial state: $|0\cdots 00\cdots 0\rangle$
- After the 1st layer: $\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|0\cdots 0\rangle\,|x\rangle$
- After the 2nd layer: $\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|f(x)\rangle\,|x\rangle$
- After the 3rd layer:

$$\frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}\,|f(x)\rangle\,|y\rangle$$

Simon's Algorithm

Quantum algorithm (Simon's algorithm)



- Initial state: $|0 \cdots 00 \cdots 0\rangle$
- After the 1st layer: $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |0 \cdots 0\rangle |x\rangle$
- After the 2nd layer: $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |f(x)\rangle |x\rangle$
- After the 3rd layer:

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |f(x)\rangle |y\rangle$$

- Probability of the result being $r \in \{0, 1\}^n$:

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} |f(x)\rangle \right|^2$$

Quantum algorithm (Simon's algorithm)

$$
\begin{aligned}
\mathcal{P}(r) &= \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} \left| f(x) \right\rangle \right|^2 \\
&= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} \left| y \right\rangle \right|^2 \\
&= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2
\end{aligned}
$$

Simon's Algorithm

Quantum algorithm (Simon's algorithm)

Case $s = 0 \cdots 0$

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} |f(x)\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} |y\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2$$

Simon's Algorithm

Quantum algorithm (Simon's algorithm)

$$
\begin{aligned}
\mathcal{P}(r) &= \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} \left| f(x) \right\rangle \right|^2 \\
&= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} \left| y \right\rangle \right|^2 \\
&= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2
\end{aligned}
$$

Case $s = 0 \cdots 0$

- f is an one-to-one function in this case.

Quantum algorithm (Simon's algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} |f(x)\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} |y\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2$$

Case $s = 0 \cdots 0$

- f is an one-to-one function in this case.
- $\mathcal{P}(r) =$

Simon's Algorithm

Quantum algorithm (Simon's algorithm)

$$
\begin{aligned}
\mathcal{P}(r) &= \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} |f(x)\rangle \right|^2 \\
&= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} |y\rangle \right|^2 \\
&= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2
\end{aligned}
$$

Case $s = 0 \cdots 0$

- f is an one-to-one function in this case.
- $\mathcal{P}(r) = \frac{1}{2^n}$

Simon's Algorithm

Quantum algorithm (Simon's algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} |f(x)\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} |y\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2$$

Case $s = 0 \cdots 0$

- f is an one-to-one function in this case.
- $\mathcal{P}(r) = \frac{1}{2^n}$

Case $s \neq 0 \cdots 0$

Quantum algorithm (Simon's algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} |f(x)\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} |y\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2$$

Case $s = 0 \cdots 0$

- f is an one-to-one function in this case.
- $\mathcal{P}(r) = \frac{1}{2^n}$

Case $s \neq 0 \cdots 0$

- f is a two-to-one function in this case.

Simon's Algorithm

Quantum algorithm (Simon's algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} |f(x)\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} |y\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2$$

Case $s = 0 \cdots 0$

- f is an one-to-one function in this case.
- $\mathcal{P}(r) = \frac{1}{2^n}$

Case $s \neq 0 \cdots 0$

- f is a two-to-one function in this case.
- $\mathcal{P}(r) =$

Simon's Algorithm

Quantum algorithm (Simon's algorithm)

$$\mathcal{P}(r) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} |f(x)\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} |y\rangle \right|^2$$

$$= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2$$

Case $s = 0 \cdots 0$

- f is an one-to-one function in this case.
- $\mathcal{P}(r) = \frac{1}{2^n}$

Case $s \neq 0 \cdots 0$

- f is a two-to-one function in this case.
- $\mathcal{P}(r) = \begin{cases} \frac{1}{2^{n-1}} & \text{if } s \cdot r = 0 \\ 0 & \text{otherwise} \end{cases}$

Simon's Algorithm

Quantum algorithm (Simon's algorithm)

$$
\begin{aligned}
\mathcal{P}(r) &= \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot r} \left| f(x) \right\rangle \right|^2 \\
&= \frac{1}{2^{2n}} \left| \sum_{y \in \{0,1\}^m} \sum_{x : f(x) = y} (-1)^{x \cdot r} \left| y \right\rangle \right|^2 \\
&= \frac{1}{2^{2n}} \sum_{y \in \{0,1\}^m} \left| \sum_{x : f(x) = y} (-1)^{x \cdot r} \right|^2
\end{aligned}
$$

Case $s = 0 \cdots 0$

- f is an one-to-one function in this case.
- $\mathcal{P}(r) = \frac{1}{2^n}$

Case $s \neq 0 \cdots 0$

- f is a two-to-one function in this case.
- $\mathcal{P}(r) = \begin{cases} \frac{1}{2^{n-1}} & \text{if } s \cdot r = 0 \\ 0 & \text{otherwise} \end{cases}$

Note that in both cases, we're picking r with $s \cdot r = 0$ uniformly at random.

Simon's Algorithm

- We use the above circuit to find a set of linearly independent set of bitstrings $r_{n-2}, \cdots, r_0$ with $r_i \cdot s = 0$.

Simon's Algorithm

- We use the above circuit to find a set of linearly independent set of bitstrings $r_{n-2}, \cdots, r_0$ with $r_i \cdot s = 0$.
- The probability that a random set of $n-1$ bitstring $r_{n-2}, \cdots, r_0$ is linearly independent is at least
$$\prod_{k=1}^{\infty}\left(1-\frac{1}{2^k}\right) \approx 0.288788$$

Simon's Algorithm

- We use the above circuit to find a set of linearly independent set of bitstrings $r_{n-2}, \cdots, r_0$ with $r_i \cdot s = 0$.
- The probability that a random set of $n-1$ bitstring $r_{n-2}, \cdots, r_0$ is linearly independent is at least

$$\prod_{k=1}^{\infty} \left(1 - \frac{1}{2^k}\right) \approx 0.288788$$

- Therefore, we need to run the circuit at most $(n-1)/0.288788 < 4n$ times on average to obtain such set of bitstrings.

Simon's Algorithm

- We use the above circuit to find a set of linearly independent set of bitstrings $r_{n-2}, \cdots, r_0$ with $r_i \cdot s = 0$.
- The probability that a random set of $n-1$ bitstring $r_{n-2}, \cdots, r_0$ is linearly independent is at least
$$\prod_{k=1}^{\infty}\left(1 - \frac{1}{2^k}\right) \approx 0.288788$$
- Therefore, we need to run the circuit at most $(n-1)/0.288788 < 4n$ times on average to obtain such set of bitstrings.
- We can use any method of our choice (such as guassian elimination) to find a non-zero bitstring t such that $r_i \cdot t = 0$ for all i.

Simon's Algorithm

- We use the above circuit to find a set of linearly independent set of bitstrings $r_{n-2}, \cdots, r_0$ with $r_i \cdot s = 0$.
- The probability that a random set of $n-1$ bitstring $r_{n-2}, \cdots, r_0$ is linearly independent is at least

$$\prod_{k=1}^{\infty} \left(1 - \frac{1}{2^k}\right) \approx 0.288788$$

- Therefore, we need to run the circuit at most $(n-1)/0.288788 < 4n$ times on average to obtain such set of bitstrings.
- We can use any method of our choice (such as guassian elimination) to find a non-zero bitstring t such that $r_i \cdot t = 0$ for all i.
- If $f(0 \cdots 0) = f(t)$, we know that $s = t$.

Simon's Algorithm

- We use the above circuit to find a set of linearly independent set of bitstrings $r_{n-2}, \cdots, r_0$ with $r_i \cdot s = 0$.
- The probability that a random set of $n-1$ bitstring $r_{n-2}, \cdots, r_0$ is linearly independent is at least
$$\prod_{k=1}^{\infty}\left(1-\frac{1}{2^k}\right) \approx 0.288788$$
- Therefore, we need to run the circuit at most $(n-1)/0.288788 < 4n$ times on average to obtain such set of bitstrings.
- We can use any method of our choice (such as guassian elimination) to find a non-zero bitstring t such that $r_i \cdot t = 0$ for all i.
- If $f(0 \cdots 0) = f(t)$, we know that $s = t$.
- Otherwise, f must be one-to-one, so $s = 0 \cdots 0$.

Summary of results for the Simon's problem

Model	Classical (Deterministic)	Classical (Probabilistic)	Quantum
Cost	$\Theta(\sqrt{2^n})$	$\Theta(\sqrt{2^n})$	$\Theta(n)$

The End