Quantum Algorithms tutorial 4

Grover's algorithm and the quantum Fourier transform

Casper Gyurik

Leiden Institute of Advanced Computer Science
October 7th, 2019

Universiteit Leiden
The Netherlands

Grover's algorithm

Quantum Fourier Transform

The unordered search problem

Given oracle access to some function $f : \{0,1\}^n \to \{0,1\}$, find a *marked* bitstring, that is, find a $j \in \{0,1\}^n$ such that $f(j) = 1$.

The unordered search problem

Given oracle access to some function $f : \{0,1\}^n \to \{0,1\}$, find a *marked* bitstring, that is, find a $j \in \{0,1\}^n$ such that $f(j) = 1$.

- This problem may be viewed as a simplification of searching an 2^n-slot unordered database.

The unordered search problem

Given oracle access to some function $f : \{0,1\}^n \to \{0,1\}$, find a *marked* bitstring, that is, find a $j \in \{0,1\}^n$ such that $f(j) = 1$.

- This problem may be viewed as a simplification of searching an 2^n-slot unordered database.
- We care about *query complexity*, i.e., how many times do we have to query the oracle to find some marked bitstring $j \in \{0,1\}^n$.

The unordered search problem

Given oracle access to some function $f : \{0,1\}^n \to \{0,1\}$, find a *marked* bitstring, that is, find a $j \in \{0,1\}^n$ such that $f(j) = 1$.

- This problem may be viewed as a simplification of searching an 2^n-slot unordered database.
- We care about *query complexity*, i.e., how many times do we have to query the oracle to find some marked bitstring $j \in \{0,1\}^n$.
- Classically, a randomized algorithm would need $\Theta(2^n)$ queries to solve the search problem.

The unordered search problem

Given oracle access to some function $f : \{0,1\}^n \to \{0,1\}$, find a *marked* bitstring, that is, find a $j \in \{0,1\}^n$ such that $f(j) = 1$.

- This problem may be viewed as a simplification of searching an 2^n-slot unordered database.
- We care about *query complexity*, i.e., how many times do we have to query the oracle to find some marked bitstring $j \in \{0,1\}^n$.
- Classically, a randomized algorithm would need $\Theta(2^n)$ queries to solve the search problem.
- Grover's algorithm solves it in $O(\sqrt{2^n})$ queries.

Grover's algorithm

The algorithm

- Recall that the "phase-oracle" was given by

$$O_{f,\pm} : |j\rangle \to (-1)^{f(j)} |j\rangle \, .$$

Grover's algorithm

The algorithm

- Recall that the "phase-oracle" was given by

$$O_{f,\pm} : |j\rangle \to (-1)^{f(j)} |j\rangle \,.$$

- Let R be the unitary transformation that maps

$$|0^n\rangle \mapsto |0^n\rangle \text{ and } |j\rangle \mapsto -|j\rangle \text{ for } j \neq 0^n.$$

Grover's algorithm

The algorithm

- Recall that the "phase-oracle" was given by

$$O_{f,\pm} : |j\rangle \to (-1)^{f(j)} |j\rangle .$$

- Let R be the unitary transformation that maps

$$|0^n\rangle \mapsto |0^n\rangle \text{ and } |j\rangle \mapsto -|j\rangle \text{ for } j \neq 0^n.$$

Grover's algorithm (known number of marked elements)

Suppose we know that $t \leq 2^n$ bitstrings are marked.

1. Set up the starting state $|U\rangle = H^{\otimes n} |0^n\rangle$.
2. Repeat the following $k = O(\sqrt{2^n/k})$ times.
   2.1 Reflect through $|B\rangle = \frac{1}{\sqrt{2^n - t}} \sum_{j \in \{0,1\}^n | f(j)=0} |j\rangle$ (i.e., apply $O_{f,\pm}$).
   2.2 Reflect through $|U\rangle$ (i.e., apply $H^{\otimes n} R H^{\otimes n}$).
3. Measure the first register and check that the resulting j is a solution.

Grover's algorithm

Why does it work?

Down the line, Grover's algorithm is about applying the "Grover iterate" $\mathcal{G}$ the correct number of times.

$$\mathcal{G} = H^{\otimes n} R H^{\otimes n} O_{f,\pm}$$

Note: the correct number of Grover iterates depends on the number of marked bitstring t (you can "overcook").

Grover's algorithm

Why does it work?

Down the line, Grover's algorithm is about applying the "Grover iterate" $\mathcal{G}$ the correct number of times.

$$\mathcal{G} = H^{\otimes n} R H^{\otimes n} O_{f,\pm}$$

Note: the correct number of Grover iterates depends on the number of marked bitstring t (you can "overcook").

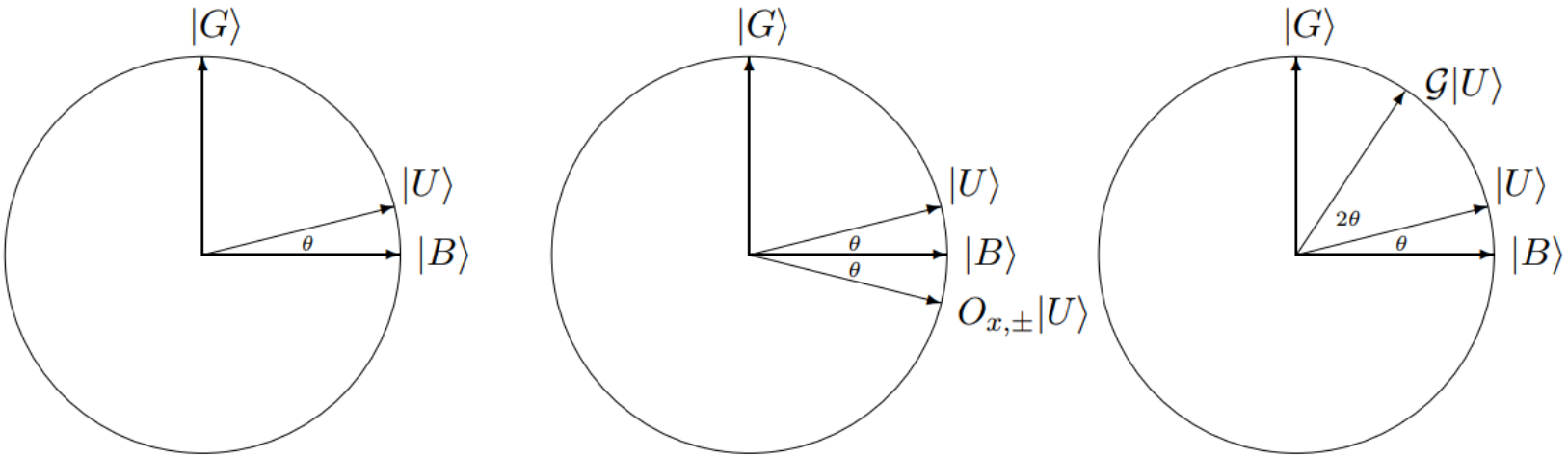

Figure: from Quantum Computing: Lecture Notes (Ronald de Wolf)

Grover's algorithm

Why does it work?

Down the line, Grover's algorithm is about applying the "Grover iterate" $\mathcal{G}$ the correct number of times.

$$\mathcal{G} = H^{\otimes n} R H^{\otimes n} O_{f,\pm}$$

Note: the correct number of Grover iterates depends on the number of marked bitstring t (you can "overcook").

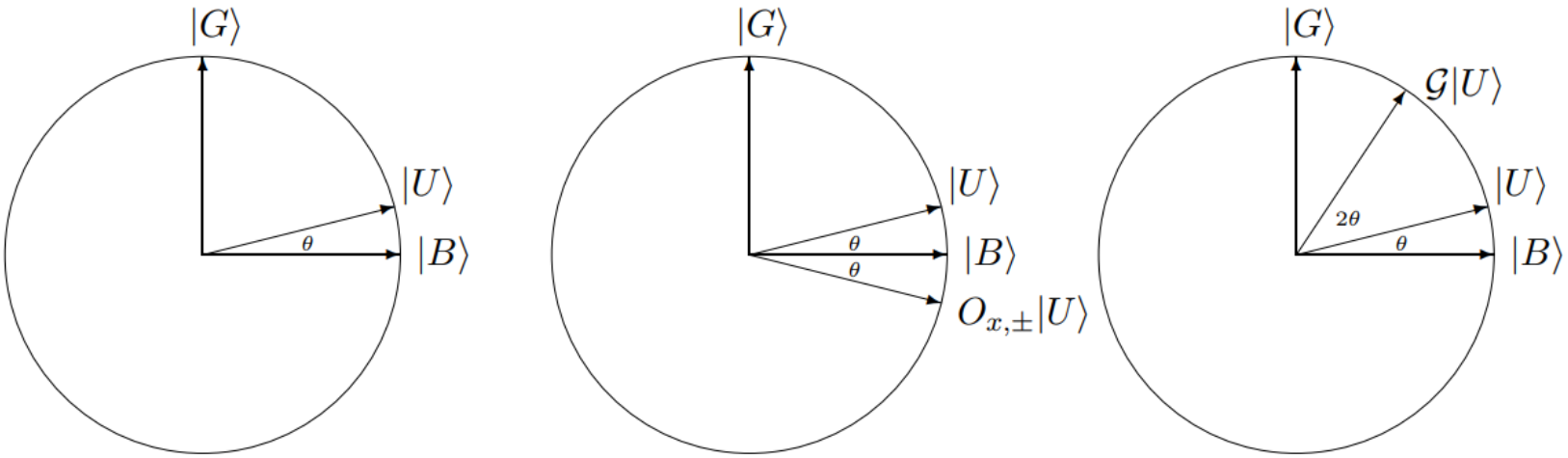

Figure: from Quantum Computing: Lecture Notes (Ronald de Wolf)

Thm: $O(\sqrt{2^n/t})$ brings you close enough to $|G\rangle = \frac{1}{\sqrt{t}} \sum_{j \in \{0,1\}^n | f(j)=1} |j\rangle$.

Grover's algorithm

Why does it work?

Down the line, Grover's algorithm is about applying the "Grover iterate" $\mathcal{G}$ the correct number of times.

$$\mathcal{G} = H^{\otimes n} R H^{\otimes n} O_{f,\pm}$$

Note: the correct number of Grover iterates depends on the number of marked bitstring t (you can "overcook").

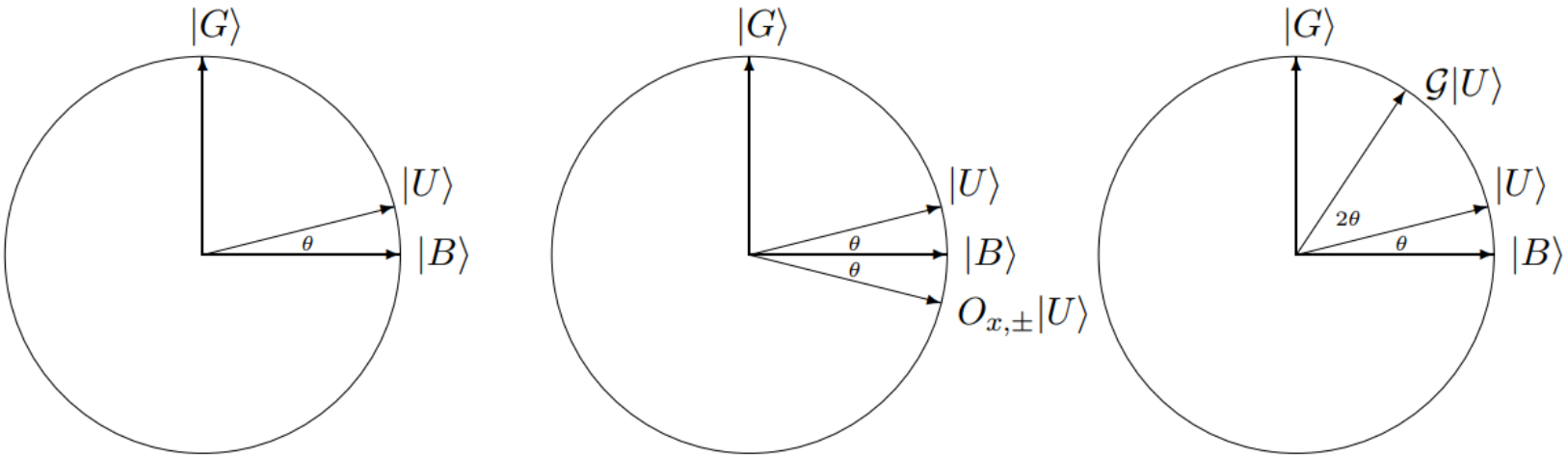

Figure: from Quantum Computing: Lecture Notes (Ronald de Wolf)

Thm: $O(\sqrt{2^n/t})$ brings you close enough to $|G\rangle = \frac{1}{\sqrt{t}} \sum_{j \in \{0,1\}^n | f(j)=1} |j\rangle$.
Note: Can also be made to work without knowing number of marked bitstrings by cleverly "guessing" and trying.

Quantum Fourier Transform

A unitary transformation

The Fourier transform

The Fourier transform is a unitary linear transformation that maps

$$F_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} \omega_{2^n}^{\mathsf{dec}(j)\mathsf{dec}(k)} |k\rangle ,$$

where $\omega_{2^n} = e^{2\pi i/2^n}$ is the 2^n-th root of unity (i.e, $\omega_{2^n}^{2^n} = 1$).

Quantum Fourier Transform

A unitary transformation

The Fourier transform

The Fourier transform is a unitary linear transformation that maps

$$F_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} \omega_{2^n}^{\mathsf{dec}(j)\mathsf{dec}(k)} |k\rangle ,$$

where $\omega_{2^n} = e^{2\pi i/2^n}$ is the 2^n-th root of unity (i.e, $\omega_{2^n}^{2^n} = 1$).

- The Fourier transform is everywhere: from signal-processing to data compression to integer multiplication.

Quantum Fourier Transform

A unitary transformation

The Fourier transform

The Fourier transform is a unitary linear transformation that maps

$$F_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} \omega_{2^n}^{\mathsf{dec}(j)\mathsf{dec}(k)} |k\rangle ,$$

where $\omega_{2^n} = e^{2\pi i/2^n}$ is the 2^n-th root of unity (i.e, $\omega_{2^n}^{2^n} = 1$).

- The Fourier transform is everywhere: from signal-processing to data compression to integer multiplication.
- Can we construct an efficient quantum circuit for the Fourier transform?

Quantum Fourier Transform

A unitary transformation

The Fourier transform

The Fourier transform is a unitary linear transformation that maps

$$F_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} \omega_{2^n}^{\mathsf{dec}(j)\mathsf{dec}(k)} |k\rangle ,$$

where $\omega_{2^n} = e^{2\pi i/2^n}$ is the 2^n-th root of unity (i.e, $\omega_{2^n}^{2^n} = 1$).

- The Fourier transform is everywhere: from signal-processing to data compression to integer multiplication.
- Can we construct an efficient quantum circuit for the Fourier transform?
- Yes, the trick is writing image of basis states as a Kronecker product state.

Quantum Fourier Transform

Writing the result as a Kronecker product state

Let us rewrite the image of the basis states as Kronecker products.
Note that for integers $k = \text{dec}(k_1 \dots k_n)$ we can write $k/2^n = \sum_{l=1}^{n} k_l 2^{-l}$.

Quantum Fourier Transform

Writing the result as a Kronecker product state

Let us rewrite the image of the basis states as Kronecker products.
Note that for integers $k = \mathsf{dec}(k_1 \dots k_n)$ we can write $k/2^n = \sum_{l=1}^{n} k_l 2^{-l}$.

$$
\begin{aligned}
F_{2^n} |j\rangle &= \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} \omega_{2^n}^{\mathsf{dec}(j)\mathsf{dec}(k)} |k_1 \dots k_n\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} e^{2\pi i \mathsf{dec}(j)\mathsf{dec}(k)/2^n} |k_1 \dots k_n\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} e^{2\pi i \mathsf{dec}(j) \sum_{l=1}^{n} k_l/2^l} |k_1 \dots k_n\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} \prod_{l=1}^{n} e^{2\pi i \mathsf{dec}(j) k_l/2^l} |k_1 \dots k_n\rangle \\
&= \bigotimes_{l=1}^{n} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \mathsf{dec}(j)/2^l} |1\rangle \right)
\end{aligned}
$$

Quantum Fourier Transform

Writing the result as a Kronecker product state

Let us rewrite the image of the basis states as Kronecker products.
Note that for integers $k = \mathsf{dec}(k_1 \dots k_n)$ we can write $k/2^n = \sum_{l=1}^{n} k_l 2^{-l}$.

$$
\begin{aligned}
F_{2^n} |j\rangle &= \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} \omega_{2^n}^{\mathsf{dec}(j)\mathsf{dec}(k)} |k_1 \dots k_n\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} e^{2\pi i \mathsf{dec}(j)\mathsf{dec}(k)/2^n} |k_1 \dots k_n\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} e^{2\pi i \mathsf{dec}(j) \sum_{l=1}^{n} k_l/2^l} |k_1 \dots k_n\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k \in \{0,1\}^n} \prod_{l=1}^{n} e^{2\pi i \mathsf{dec}(j) k_l/2^l} |k_1 \dots k_n\rangle \\
&= \bigotimes_{l=1}^{n} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \mathsf{dec}(j)/2^l} |1\rangle \right)
\end{aligned}
$$

Exercise: Derive the last equality for $n = 2$ qubits.

Quantum Fourier Transform

An efficient circuit

Goal: construct circuit that maps

$$|j_1 \dots j_n\rangle \mapsto \bigotimes_{l=1}^{n} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \mathsf{dec}(j)/2^l} |1\rangle \right).$$

Quantum Fourier Transform

An efficient circuit

Goal: construct circuit that maps

$$|j_1 \dots j_n\rangle \mapsto \bigotimes_{l=1}^{n} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \mathsf{dec}(j)/2^l} |1\rangle \right).$$

So, each qubit $|j_l\rangle$ has to be mapped to $\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \mathsf{dec}(j)/2^l} |1\rangle \right)$.

Quantum Fourier Transform

An efficient circuit

Goal: construct circuit that maps

$$|j_1 \dots j_n\rangle \mapsto \bigotimes_{l=1}^{n} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \mathrm{dec}(j)/2^l} |1\rangle \right).$$

So, each qubit $|j_l\rangle$ has to be mapped to $\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \mathrm{dec}(j)/2^l} |1\rangle \right)$.

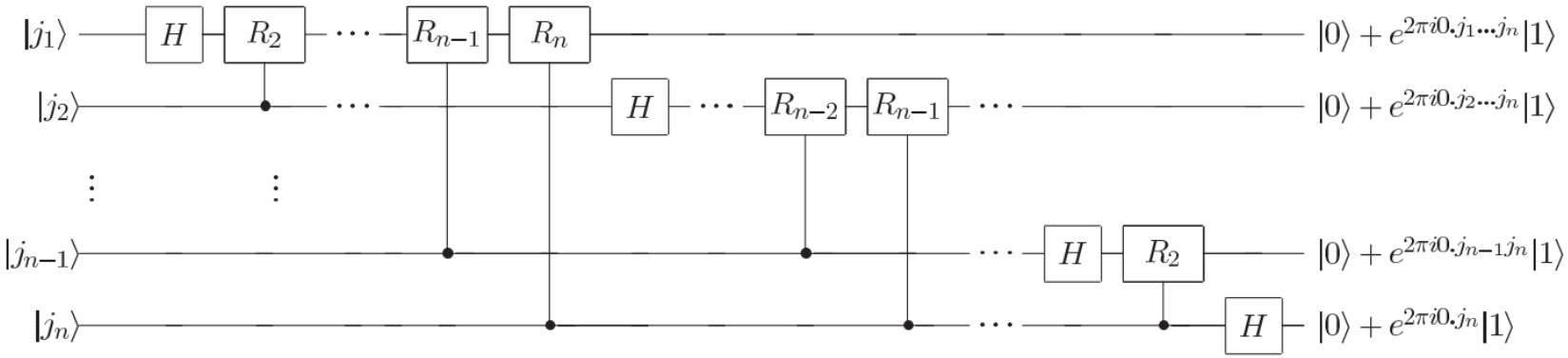

Figure: From Nielsen & Chuang.

Exercise: confirm correctness of the above circuit for $n = 2$ qubits.