Week 6: Deutsch's and Simon's Algorithm

COMS 4281 (Fall 2025)

Admin

1. Pset1 due Friday, October 10, 11:59pm.
2. Midterm on Thursday, October 16. You will be assigned an exam room (either Havemeyer 209 or Hamilton 703).
3. No worksheet this week (focus on the pset); long practice worksheet released at the end of the week.

Quantum seminar bonanza!

Thursday (Oct 9)

1. 3pm in CSB 453: Jin-Yi Cai on “Shor’s Quantum Algorithms Fail in the Presence of Noise” (CS Theory Seminar)
2. 4pm in Italian Academy: Scott Aaronson on “Computational Complexity and Explanations in Physics” (Patrick Suppes Lecture in Philosophy)

Friday (Oct 10):

1. 12:30pm in CSB 453: Scott Aaronson on “New Results on Quantum Oracles” (CS Theory Seminar)

Last time: power of entanglement, and quantum circuits

- EPR Paradox and Bell's theorem
- Quantum circuit model, universal gate set

Our first quantum algorithm

Deutsch's problem

Given oracle access to a boolean function $f : \{0, 1\} \to \{0, 1\}$, decide whether $f(0) = f(1)$ or $f(0) \neq f(1)$.

Deutsch's problem

Given oracle access to a boolean function $f : \{0, 1\} \to \{0, 1\}$, decide whether $f(0) = f(1)$ or $f(0) \neq f(1)$.

Oracle access to a function f means that the computer can only access it as a **black-box**, i.e., query some inputs, and get the corresponding outputs. The computer cannot access how the black box is implemented.

Deutsch's problem

Given oracle access to a boolean function $f : \{0, 1\} \to \{0, 1\}$, decide whether $f(0) = f(1)$ or $f(0) \neq f(1)$.

Oracle access to a function f means that the computer can only access it as a **black-box**, i.e., query some inputs, and get the corresponding outputs. The computer cannot access how the black box is implemented.

Claim: Any classical algorithm that solves the Deutsch problem must make 2 queries to f.

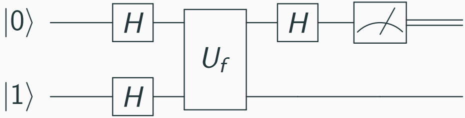
Quantum oracles

Quantum algorithms can access a black-box function f through a unitary U_f corresponding to the **reversible** version of f. For $f : \{0,1\} \to \{0,1\}$, this is a two-qubit unitary

$$U_f \left|x, b\right\rangle = \left|x, b \oplus f(x)\right\rangle .$$

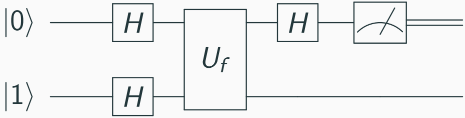
A quantum circuit that wants to access f will simply call U_f just like any other two-qubit gate.

Deutsch's algorithm



This quantum algorithm solves the Deutsch problem with **one** call to U_f.

Deutsch's algorithm



This quantum algorithm solves the Deutsch problem with **one** call to U_f.

Let's do this on the board!

Deutsch's algorithm

The algorithm evaluates the function f in **superposition**. This seems to give a 2x speedup!

Is this cheating? Maybe the "quantum access" is just really making multiple classical queries under the hood?

Observation: the qubit storing the answer at the end corresponds to the *input wire* of the oracle U_f. We don't care about the output wire!

This is a common feature in many quantum algorithms with exponential speedup.

Simon’s Problem

Oracles with multiple output bits

Often we consider oracles for functions $f : \{0,1\}^n \to \{0,1\}^t$ with *multiple output bits*.

Oracles with multiple output bits

Often we consider oracles for functions $f : \{0,1\}^n \to \{0,1\}^t$ with *multiple output bits*.

A quantum circuit can query such functions f using the $(n+t)$-qubit unitary

$$U_f \,|\, \underbrace{x}_{n \text{ qubits}}, \underbrace{b}_{t \text{ qubits}} \rangle = |x, b \oplus f(x)\rangle$$

Oracles with multiple output bits

Often we consider oracles for functions $f : \{0,1\}^n \to \{0,1\}^t$ with *multiple output bits*.

A quantum circuit can query such functions f using the $(n+t)$-qubit unitary

$$U_f \,|\underbrace{x}_{n \text{ qubits}}, \underbrace{b}_{t \text{ qubits}}\rangle = |x, b \oplus f(x)\rangle$$

Here $b \oplus f(x)$ denotes the **bitwise** XOR of the strings b and $f(x)$, e.g., if $b = 010$ and $f(x) = 110$ then

$$b \oplus f(x) = 100 \ .$$

Simons Problem

Problem: Given oracle access to $f : \{0,1\}^n \rightarrow \{0,1\}^n$ such that there exists a nonzero **secret string** $s \in \{0,1\}^n$ where for all $x, y \in \{0,1\}^n$

$$f(x) = f(y) \Leftrightarrow x \oplus y = s$$

find the secret string s.

Simons Problem

Example function f:

x	$f(x)$
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

What's the secret s?

Simons Problem

Problem: Given oracle access to $f : \{0,1\}^n \to \{0,1\}^n$ such that there exists a nonzero **secret string** $s \in \{0,1\}^n$ where for all $x, y \in \{0,1\}^n$

$$f(x) = f(y) \Leftrightarrow x \oplus y = s$$

find the secret string s.

Question: How many queries to f are needed to find the secret?

Classical algorithm to solve Simons Problem

1. Randomly sample $x_1, \ldots, x_K \in \{0,1\}^n$ for $K = 10\sqrt{2^n}$.
2. Check if there exists a pair $x_i \neq x_j$ where $f(x_i) = f(x_j)$. If so, then output $s = x_i \oplus x_j$.

Classical algorithm to solve Simons Problem

1. Randomly sample $x_1, \ldots, x_K \in \{0,1\}^n$ for $K = 10\sqrt{2^n}$.
2. Check if there exists a pair $x_i \neq x_j$ where $f(x_i) = f(x_j)$. If so, then output $s = x_i \oplus x_j$.

By the **birthday paradox**, this algorithm will find the secret with high probability. Requires $O(2^{n/2})$ queries to f.

(Show on board)

Classical algorithm to solve Simons Problem

1. Randomly sample $x_1, \ldots, x_K \in \{0,1\}^n$ for $K = 10\sqrt{2^n}$.
2. Check if there exists a pair $x_i \neq x_j$ where $f(x_i) = f(x_j)$. If so, then output $s = x_i \oplus x_j$.

By the **birthday paradox**, this algorithm will find the secret with high probability. Requires $O(2^{n/2})$ queries to f.

(Show on board)

$2^{n/2}$ queries are necessary for any classical algorithm!

Simons Algorithm

Simons algorithm is a **classical-quantum** hybrid algorithm.

It uses the quantum computer as a *subroutine* to sample from a distribution many times, and uses **classical post-processing** to extract the secret.

Simons Algorithm

Simons algorithm is a **classical-quantum** hybrid algorithm.

It uses the quantum computer as a *subroutine* to sample from a distribution many times, and uses **classical post-processing** to extract the secret.

Simons subroutine: Quantum circuit queries U_f *once* and obtains a *uniformly random* string $y \in \{0,1\}^n$ where the inner product of y and the secret s,

$$s \cdot y = s_1 y_1 \oplus s_2 y_2 \oplus \cdots \oplus s_n y_n$$

is equal to 0.

Simons algorithm, classical post-processing

Classical post-processing: Obtain $m = 100n$ samples $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$ such that

$$\begin{gathered} y^{(1)} \cdot s = 0 \\ y^{(2)} \cdot s = 0 \\ \vdots \\ y^{(m)} \cdot s = 0 \end{gathered}$$

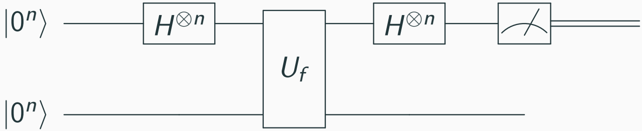
Simons algorithm, classical post-processing

Classical post-processing: Obtain $m = 100n$ samples $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$ such that

$$\begin{gathered} y^{(1)} \cdot s = 0 \\ y^{(2)} \cdot s = 0 \\ \vdots \\ y^{(m)} \cdot s = 0 \end{gathered}$$

With high probability, can solve this system of linear equations using Gaussian elimination to get s.

Simons subroutine



$H^{\otimes n}$ means applying H to n qubits independently.

Hadamard math

We know that $H\left|0\right\rangle = \left|+\right\rangle$. What is $H^{\otimes n}\left|0\right\rangle^{\otimes n}$?

Hadamard math

We know that $H\left|0\right\rangle = \left|+\right\rangle$. What is $H^{\otimes n}\left|0\right\rangle^{\otimes n}$?

$$H^{\otimes n}\left|0\right\rangle^{\otimes n} = \left(H\left|0\right\rangle\right)^{\otimes n} = \left|+\right\rangle^{\otimes n} \ .$$

Hadamard math

We know that $H\left|0\right\rangle = \left|+\right\rangle$. What is $H^{\otimes n}\left|0\right\rangle^{\otimes n}$?

$$H^{\otimes n}\left|0\right\rangle^{\otimes n} = (H\left|0\right\rangle)^{\otimes n} = \left|+\right\rangle^{\otimes n} \ .$$

This in turn is

$$\left|+\right\rangle^{\otimes n} = \left(\frac{1}{\sqrt{2}}\left|0\right\rangle + \frac{1}{\sqrt{2}}\left|1\right\rangle\right)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} \left|x_1, \ldots, x_n\right\rangle$$

Hadamard math

Fix $x_1, \ldots, x_n \in \{0, 1\}$. What is $H^{\otimes n} |x_1, \ldots, x_n\rangle$?

Hadamard math

Fix $x_1, \ldots, x_n \in \{0, 1\}$. What is $H^{\otimes n} |x_1, \ldots, x_n\rangle$?

This is

$$(H|x_1\rangle) \otimes (H|x_2\rangle) \otimes \cdots \otimes (H|x_n\rangle)$$
$$= \frac{1}{\sqrt{2^n}} \Big(|0\rangle + (-1)^{x_1} |1\rangle \Big) \otimes \Big(|0\rangle + (-1)^{x_2} |1\rangle \Big) \otimes \cdots \otimes \Big(|0\rangle + (-1)^{x_n} |1\rangle \Big)$$

Hadamard math

Fix $x_1, \ldots, x_n \in \{0, 1\}$. What is $H^{\otimes n} |x_1, \ldots, x_n\rangle$?

This is

$$
\begin{aligned}
&(H|x_1\rangle) \otimes (H|x_2\rangle) \otimes \cdots \otimes (H|x_n\rangle) \\
&= \frac{1}{\sqrt{2^n}}\Big(|0\rangle + (-1)^{x_1}|1\rangle\Big) \otimes \Big(|0\rangle + (-1)^{x_2}|1\rangle\Big) \otimes \cdots \otimes \Big(|0\rangle + (-1)^{x_n}|1\rangle\Big) \\
&= \frac{1}{\sqrt{2^n}} \sum_{y_1, y_2, \ldots, y_n \in \{0,1\}} (-1)^{x_1 y_1}|y_1\rangle \cdots (-1)^{x_n y_n}|y_n\rangle
\end{aligned}
$$

Hadamard math

Fix $x_1, \ldots, x_n \in \{0,1\}$. What is $H^{\otimes n} |x_1, \ldots, x_n\rangle$?

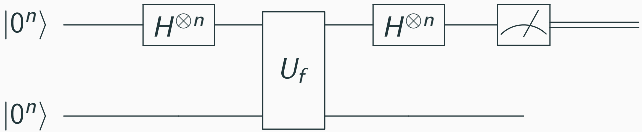
This is

$$
\begin{aligned}
&(H|x_1\rangle) \otimes (H|x_2\rangle) \otimes \cdots \otimes (H|x_n\rangle) \\
&= \frac{1}{\sqrt{2^n}} \Big(|0\rangle + (-1)^{x_1} |1\rangle \Big) \otimes \Big(|0\rangle + (-1)^{x_2} |1\rangle \Big) \otimes \cdots \otimes \Big(|0\rangle + (-1)^{x_n} |1\rangle \Big) \\
&= \frac{1}{\sqrt{2^n}} \sum_{y_1, y_2, \ldots, y_n \in \{0,1\}} (-1)^{x_1 y_1} |y_1\rangle \cdots (-1)^{x_n y_n} |y_n\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle
\end{aligned}
$$

where $x \cdot y$ denotes the inner product of the strings x and y modulo 2:

$$x \cdot y = x_1 y_1 + \cdots + x_n y_n \mod 2\ .$$

Simons subroutine



Let's analyze this on the board.

Simons algorithm

- Makes $O(n)$ queries to U_f and solves the problem with high probability
- Once again, the valuable information is stored not in the answer register of U_f, but in the input register.

Simons algorithm

- Making crucial use of constructive/destructive interference!
- It's finding **global hidden structure** in the function.
- Is this speedup more convincing?

Simons algorithm

- Invented by Dan Simons in 1992, and was the first example of a problem that could be solved exponentially faster with a quantum algorithm compared to a classical randomized algorithm.
- This algorithm directly inspired Peter Shor to invent the famous factoring algorithm.
- Recently, Simons algorithm also has applications to breaking symmetric key cryptography.