Last class we discussed the black-box model and quantum query complexity with our first example of a truly quantum algorithm—Deutsch's algorithm. Today we continue with query algorithms, adding more complexity to our functions and the interference patterns leading to the desired answer. Just remember:

Quantum algorithms =
Superposition, interference, & entanglement
The next quantum algorithm we're going to see is a straightforward generalization of Deutsch's algorithm to the case when \( f \) takes \( n \) (rather than 1) inputs.

Let \( f : \{0,1\}^n \rightarrow \{0,1\} \). We say:

1. \( f \) is constant if \( f(x) = f(y) \) \( \forall x, y \in \{0,1\}^n \)
2. \( f \) is balanced if \( \sum_{x \in \{0,1\}^n} f(x) = 0 \) for the other half.

**Deutsch-Jozsa's problem (DJ)**

**Input:** a function \( f : \{0,1\}^n \rightarrow \{0,1\} \)

**Promise:** \( f \) is either constant or balanced

**Goal:** Determine whether \( f \) is constant or balanced

**Fact:** The classical query complexity is \( 2^{n-1} + 1 \)

\( \rightarrow \) Why? Suppose the first \( 2^{n-1} \) queries (i.e. half the strings \( x \in \{0,1\}^n \))
give \( f(x) = 0 \). Then the other half of the strings could either all give 0 — hence \( f \) is constant — or could all give 1 — hence \( f \) is balanced.

Deutsch & Jozsa showed that the quantum query complexity of their problem is one!
The Deutsch-Jozsa algorithm works analogously to Deutsch's algorithm, but with \( n \) qubits.

![Diagram](image)

**Uniform superposition**

The first stage of the DJ algorithm is so common and important it deserves a separate analysis.

Consider the circuit

![Circuit Diagram](image)

The state this circuit prepares is

\[
(\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)) \otimes (\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle))
\]

\[= \frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle)
\]

\[= \frac{1}{2} \sum_{x \in \{0, 1\}^n} 1|x\rangle
\]

This is a uniform superposition of \( x \in \{0, 1\}^n \).

In general,

\[
H \otimes^n 10 \otimes^n = (H \otimes H \otimes \ldots \otimes H) (|0\rangle \otimes |0\rangle \otimes \ldots \otimes |0\rangle)
\]

\[= (\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)) \otimes^n
\]

\[= (\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)) \otimes^n
\]

\[= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0, 1\}^n} 1|x\rangle
\]
So, Deutsch-Jozsa first prepares the uniform superposition then uses $U_f |x\rangle = (-1)^{f(x)} |x\rangle$ to phase each string:

$$U_f \otimes^n |0\rangle^\otimes^n = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle^n$$

As in the Deutsch algorithm, the final $H^\otimes^n$ is going to generate interference.

But how?

(Hadamard gate abstractly)

Note that $H|x\rangle = \frac{1}{\sqrt{2}} (|0\rangle + (-1)^x |1\rangle)$, $x \in \{0,1\}$

We can write this more compactly as

$$H|x\rangle = \frac{1}{\sqrt{2}} (|0\rangle + (-1)^x |1\rangle) = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{z \cdot x} |z\rangle$$

Now what happens if we do this to an $n$-bit string?

$$H^\otimes^n |x_1, x_2, \ldots, x_n\rangle = \left( \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{z \cdot x_1} |z\rangle \right) \otimes \cdots \otimes \left( \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{z \cdot x_n} |z\rangle \right)$$

Note: we’ll largely start using $\mathbb{Z}_2$ (integers mod 2) to refer to $\{0,1\}$ now. If $x, y \in \mathbb{Z}_2^n$, $x \cdot y = x_1 y_1 \oplus \cdots \oplus x_n y_n = x_1 y_1 \oplus \cdots \oplus x_n y_n \mod 2$

So, the final state in the DJ algorithm is

$$H^\otimes^n \left( \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \right) = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} \left( \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{z \cdot x} |z\rangle \right)$$

$$= \frac{1}{2^n} \sum_{x, z \in \{0,1\}^n} (-1)^{f(x) + z \cdot x} |z\rangle$$
Interference analysis

The algorithm looks like this:

We need to figure out which paths interfere.
Consider a single \( z \). The amplitude of this \( z \) is the sum over all paths leading to it:

\[
\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} 100...0
\]

What is the amplitude of \( z = 00...0 \)?

**Case 1: **\( f \) is constant

Then \( \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} 100...0 = \frac{1}{2^n} \sum_x (-1)^f 100...0 \)

\[= \pm 100...0 \]

**Case 2: **\( f \) is balanced

The \( \frac{1}{2^n} \sum_x (-1)^{f(x)} 100...0 = \frac{1}{2^n} (\sum_{x \mid f(x) = 0} 100...0 + \sum_{x \mid f(x) = 1} -100...0) \)

\[= \frac{2^{-n}}{2^n} 100...0 - \frac{2^{n-1}}{2^n} 100...0 \]

\[= 0 \]

So, if we measure at the end, if \( f \) is constant we get \( 100...0 \) with 100% probability, and if \( f \) is balanced we get \( 100...0 \) with 0% probability!
The Deutsch-Jozsa algorithm is not that impressive in reality, because we can solve the problem with \(\frac{2}{3}\) probability with 2 queries classically using a randomized algorithm. Bernstein & Vazirani came up with the next algorithm that gives a non-trivial speed-up over randomized algorithms too! Their algorithm is identical to Deutsch-Jozsa, but involves a specially-chosen promise on \(f\).

**Bernstein-Vazirani problem (BV)**

- **Input**: a function \(f : \{0,1\}^n \rightarrow \{0,1\}\)
- **Promise**: \(f(x) = s \cdot x \mod 2 \quad \forall x \in \{0,1\}^n\) for some \(s \in \{0,1\}^n\)
- **Goal**: find the hidden string \(s\)

**Fact**

The probabilistic query complexity of BV is at least \(n\). Why? Because we need \(n\) bits of information and \(f\) only gives us 1 bit.

Bernstein & Vazirani's algorithm uses the exact same circuit as Deutsch & Jozsa's, but a different interference analysis.

![Circuit Diagram](diagram.png)

**Final state**: 
\[
\frac{1}{2^n} \sum_{x,z \in \{0,1\}^n} (-1)^{f(x)+x \cdot z} |12\rangle
\]
The simple analysis is, just like Deutsch-Jozsa, to look at the amplitude of a well-chosen string. This time, we'll analyze interference when $Z = 5$.

\[
\frac{1}{2^n} \sum_{x \in \{0, 1\}^n} (-1)^{f(x) + x \cdot 5} |15\rangle = \frac{1}{2^n} \sum_{x \in \{0, 1\}^n} (-1)^{5x + x \cdot 5} |15\rangle \\
= \frac{1}{2^n} \sum_{x \in \{0, 1\}^n} |15\rangle \\
= |15\rangle
\]

So measuring in the computational basis results in $5$ with 100% probability!

Simple, right?