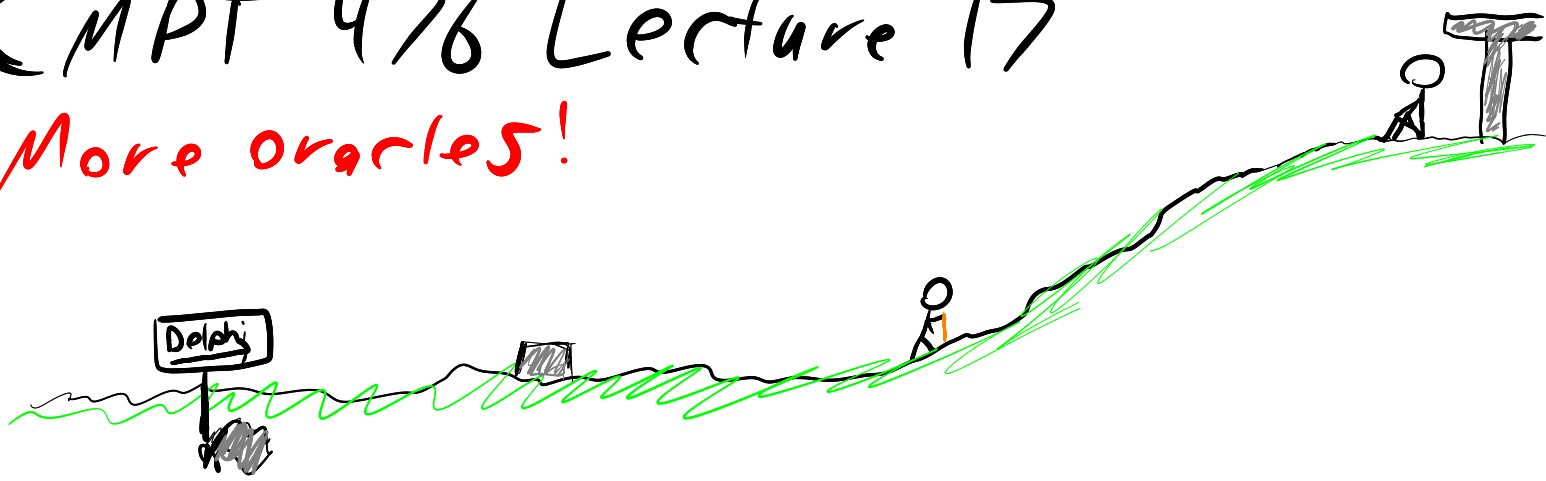


CMP 476 Lecture 17

More oracles!



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

1. Prepare superpositions of all x in $\{0, 1\}^n$
2. Phase the state by $(-1)^{f(x)}$
3. Sum up all paths (values of x)

(Deutsch-Jozsa algorithm)

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 $f(x) = 1$ for exactly half of the strings $x \in \{0,1\}^n$, and $f(x) = 0$ for the other half.

Deutsch-Jozsa's problem tries to find whether f is balanced or constant

We can make a probabilistic queries to get the right answer with a probability of $2/3$

Deutsch-Jozsa's problem (DJ)

Input: a function $f: \{0,1\}^n \rightarrow \{0,1\}$

Promise: f is either constant or balanced

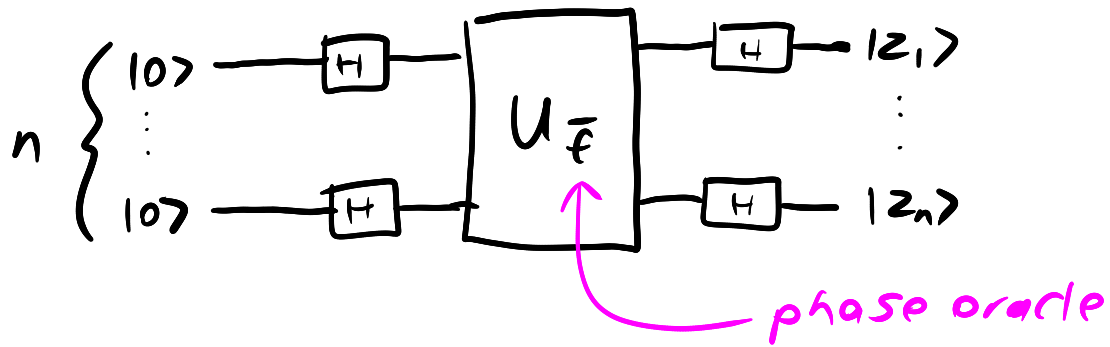
Goal: Determine whether f is constant or balanced

Fact: The classical query complexity is $2^{n-1} + 1$

↳ 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.



(Uniform superposition)

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

Consider the circuit

The state this circuit prepares is

$$\begin{aligned}
 (H|0\rangle) \otimes (H|0\rangle) &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\
 &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\
 &= \frac{1}{2} \sum_{x \in \{0,1\}^2} |x\rangle
 \end{aligned}$$

This is a uniform superposition of $x \in \{0,1\}^2$.

In general,

$$\begin{aligned}
 H^{\otimes n} |0\rangle^{\otimes n} &= \overbrace{(H \otimes H \otimes \dots \otimes H)}^{n \text{ times}} \overbrace{|0\rangle|0\rangle\dots|0\rangle}^{n \text{ times}} \\
 &= (H|0\rangle)^{\otimes n} \\
 &= \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right)^{\otimes n} \\
 &= \frac{1}{\sqrt{2}^n} \sum_{x \in \{0,1\}^n} |x\rangle
 \end{aligned}$$

Uniform superposition is obtained by applying Hadamard gate to n copies of zero.

So, Deutsch-Jozsa first prepares the uniform superposition then uses $U_f|x\rangle = (-1)^{f(x)}|x\rangle$ to phase each string:

f tilde adds a phase to $|x\rangle$ state.

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

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

the hadamard gate takes x and returns the + or - state, it encodes the relative phase

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

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

$$H^{\otimes n} |x_1, x_2, \dots, x_n\rangle = \left(\frac{1}{\sqrt{2}} \sum_z (-1)^{x_1 \cdot z_1} |z_1\rangle \right) \otimes \dots \otimes \left(\frac{1}{\sqrt{2}} \sum_{z_n} (-1)^{x_n \cdot z_n} |z_n\rangle \right)$$

If it is even, we have a phase of 1. -1 otherwise.

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

$$\begin{aligned} &= \frac{1}{\sqrt{2}^n} \sum_{z_1, \dots, z_n} (-1)^{x_1 \cdot z_1 + \dots + x_n \cdot z_n} |z_1, \dots, z_n\rangle \\ &= \frac{1}{\sqrt{2}^n} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle \end{aligned}$$

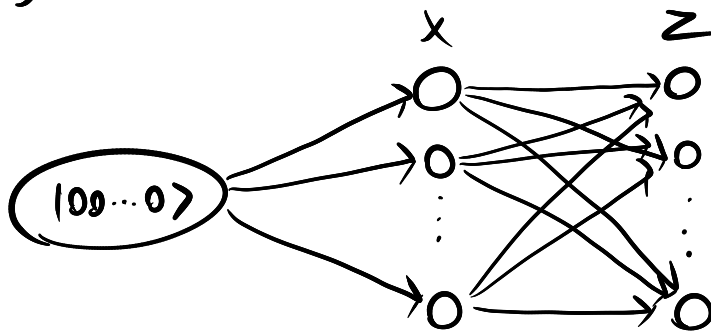
← dot product over $\{0,1\}^n = \mathbb{Z}_2^n$

So, the final state in the DJ algorithm is

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

(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:

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

Do I have 2^n possible values of x ?

What is the amplitude of $z = 00 \dots 0$?

Case 1: f is constant

All zeros as amplitude state?

$$\text{Then } \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |00 \dots 0\rangle = \frac{1}{2^n} \sum_x (-1)^c |00 \dots 0\rangle$$

How are we obtaining this amplitude?

$$= \pm |00 \dots 0\rangle$$

(Case 2: f is balanced)

2^{n-1} of both states offering no phase difference since f is balanced?

$$\begin{aligned} \text{The } \frac{1}{2^n} \sum_x (-1)^{f(x)} |00 \dots 0\rangle &= \frac{1}{2^n} \left(\sum_{x|f(x)=0} |00 \dots 0\rangle + \sum_{x|f(x)=1} -|00 \dots 0\rangle \right) \\ &= \frac{2^{n-1}}{2^n} |00 \dots 0\rangle - \frac{2^{n-1}}{2^n} |00 \dots 0\rangle \\ &= 0 \end{aligned}$$

So, if we measure at the end, if f is constant we get $|00 \dots 0\rangle$ with 100% probability, and if f is balanced we get $|00 \dots 0\rangle$ with 0% probability!

(Bernstein-Vazirani algorithm) it is a way of framing the Deutsch-Jozsa algorithm to find out informations about n bits.

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 .

Phase oracle vs state oracle:
oracle: Applies f to the bit string in the phase. Takes a bit and returns the phase of the bit. Takes $|x\rangle$ and returns $(-1)^{f(x)}|x\rangle$
State oracle: Takes a state $|x\rangle|c\rangle$ and sends it to $|x\rangle|c \oplus f(x)\rangle$

Apply f in the phase:

Bernstein-Vazirani problem (BV)

Input: a function $f: \{0,1\}^n \rightarrow \{0,1\}$

Promise: $f(x) = s \cdot x \bmod 2 \quad \forall x \in \{0,1\}^n$ for some $s \in \{0,1\}^n$

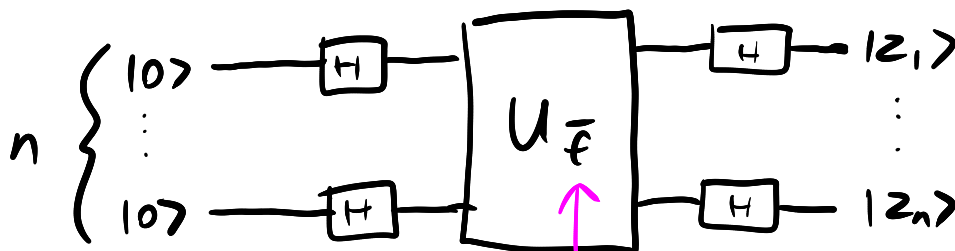
Goal: find the hidden string s

The hidden string is the definition of the function.

Fact At least n since you can imagine the case where each query correctly gives you one of the bits of s

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



We only get constructive interference for the basis state z_n .
Why do we not need to do an analysis of the other states?
All the amplitudes are in one state.

phase oracle

The $f(x)$ in the exponent comes from the phase shift.

$$\text{Final state: } \frac{1}{2^n} \sum_{x, z \in \{0,1\}^n} (-1)^{f(x) + x \cdot z} |z\rangle$$

(Interference analysis)

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 $z = S$.

$$\begin{aligned}\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) + x \cdot S} |S\rangle &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{S \cdot x + x \cdot S} |S\rangle \\ &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} |S\rangle \\ &= |S\rangle\end{aligned}$$

Ask about this.

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

Simple, right?