**6.896 Quantum Complexity Theory** Sept 23, 2008

# Lecture 6

*Lecturer: Scott Aaronson*

**Last Time:**

- Quantum Error-Correction
- Quantum Query Model
- Deutsch-Jozsa Algorithm (Computes $x \oplus y$ in one query.)

**Today:**

- Bernstein-Vazirini Algorithm
- Simon's Algorithm
- Shor's Algorithm[1]
- Hidden Subgroup Framework[2]

## 1 Recap of Last Lecture

### 1.1 Error Propagation

Recall from the previous lecture that error propagates differently in quantum computing compared with in classical computing. In particular, the (1990's) Threshold Theorem states that quantum computing is robust to small errors since it is a linear theory. This implies that small errors are not magnified over the course of the computation and can be corrected using sophisticated hierarchical error-correcting codes. This is unlike classical computing where noise amplification was a major consideration in designing digital systems.

### 1.2 The Query Model

In quantum complexity theory, we are interested in determining which problems are efficiently solvable by quantum algorithms and which ones are not. This is a difficult problem to answer since we don't know to quantify the number of required computational steps. We don't even know how to prove that the problems we care about do not require an exponential number of steps.

[1] We only introduced this.
[2] We didn't get to this.

The Query Model allows us to make some analysis of the computational complexity of quantum algorithms. In this model, the complexity of an algorithm is measured as the number of queries we make to some query box. So, what do we mean by a query? In classical computing, the idea is fairly self-explanatory. For example, to find the majority of three input bits $x_1$, $x_2$, and $x_3$, we may query for the values for $x_1$ and $x_3$. If we find that both bits are one, then we will need a total of two queries.

In quantum computing, the queries (like all computational steps) must be reversible and, therefore, unitary. In the previous lecture, we gave two equivalent models of quantum queries. In the first model, the function $f(\cdot)$ that we want to query becomes xor-ed to some answer bit $b$. In the second model, $f(\cdot)$ gets written to a phase; we flip the amplitude if the answer is one and, otherwise, we don't. Written out, the two definitions of queries are:

$$|x\rangle|b\rangle \longrightarrow |x\rangle|b \oplus f(x)\rangle \tag{1}$$

$$|x\rangle \longrightarrow (-1)^{f(x)\cdot b}|x\rangle \tag{2}$$

Last time, we talked about how Equation (1) can simulate Equation (2). We asserted that Equation (2) can also simulate Equation (1).

### 1.3 Deutsch-Jozsa Algorithm

The Deutsch-Jozsa algorithm is our first example where quantum computing is more efficient than classical computing. The algorithm computes the exclusive-or of two bits in one query rather than two. (In classical computing, we need to query both bits to find the exclusive-or.) We can extend this algorithm to exclusive-or $n$-bits using only $n/2$ queries. So, this is a factor of two speed-up.

The algorithm is the following. We query in superposition and then apply a Hadamard.

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \longrightarrow \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}$$

If $f(0)$ is equal to $f(1)$, then the Hadamard is going to map above state to $\pm|0\rangle$. Otherwise, if $f(0) \neq f(1)$, then the Hadamard is going to map to $\pm|1\rangle$.

$$\longrightarrow \pm|0\rangle, \text{ if } f(0) = f(1)$$
$$\longrightarrow \pm|1\rangle, \text{ if } f(0) \neq f(1)$$

## 2 Bernstein-Vazarani Algorithm [93]

The Deutsch-Jozsa algorithm can be generalized to the problem of finding a hidden linear structure. Suppose that there is a boolean function, $f : \{0,1\}^n \longrightarrow \{0,1\}$, which maps $n$-bit strings to a single bit. In addition, we have query access to the function. Like in the Deusch-Jozsa algorithm, we can query any $n$-bit string $x$ for $f(x)$; and we can also query in superposition.

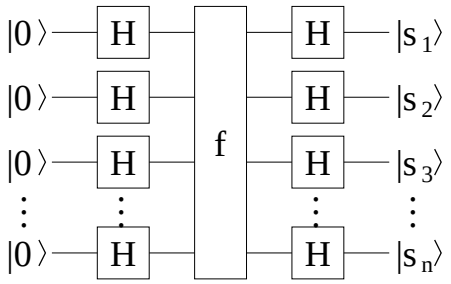
Let us consider the case where $f(x)$ is of the form $s \cdot x \mod 2$, where $s$ is a fixed and secret string. (Here "$\cdot$" denotes the inner product, i.e. $s \cdot x = s_1x_1 + ... + s_nx_n \mod 2$.) In other words, we are promised that there exists a "secret string" $s$ such that $f(x) = s \cdot x \mod 2$ for all $x$. The problem is to find $s$.

First, let us examine the query complexity in the classical world. How many queries do we need to solve this problem classically? $n$ queries are sufficient, because we can query the basis strings. For example, for $n = 5$, we can query the basis strings to find $s_1$, ..., $s_5$ as follows:

$$\begin{aligned} f(10000) &= s_1 \\ f(01000) &= s_2 \\ f(00100) &= s_3 \\ f(00010) &= s_4 \\ f(00001) &= s_5 \end{aligned}$$

In addition, $n$ is also the lower bound. We can prove that we need at least $n$ queries using a basic information theoretic argument. There are $2^n$ possibilities for $s$, and each query can only cut the space in half.

In contrast, the Bernstein-Vazarani algorithm solves the problem using only one query. The quantum circuit diagram of the Bernstein-Vazarani algorithm is given below:




Mathematically, this is equivalent to:

$$\begin{aligned} |0\rangle^{\otimes n} &\longrightarrow \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \\ &\longrightarrow \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{s\cdot x}|x\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{s_1x_1}...(-1)^{s_nx_n}|x\rangle \end{aligned}$$

After querying the box in superposition, the the secret string $s$ written in the Hadamard basis. So in order to read $s$, we convert back to standard basis. This is what is accomplished by applying a second Hadamard in the end. Notice that although we needed a linear number of gates, we only have to make one query. This is an $n$-to-1 speed-up.

# 3 Simon's Algorithm [93]

Using the Bernstein-Vazarani algorithm, we have demonstrated that we need only one query in the quantum world where we needed at $n$ in the classical world. While this is an interesting result, what we really wish for is an exponential-to-polynomial speed-up. What we want to ask is, "Can we use the quantum model to attack the Beast that is Exponentially?" This questions leads us to Simon's algorithm.[3]

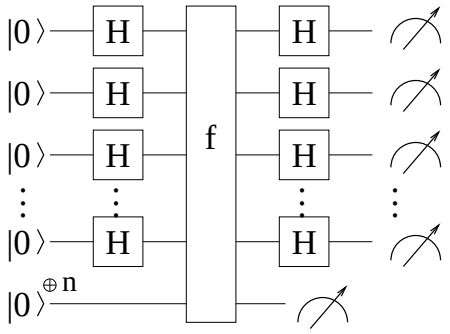
As in the Bernstein-Vazarani algorithm, we are again given a query box that computes some function $f(x)$. In this case, the function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ maps $n$-bits to $n$-bits. As before, there is a promise associated with the function $f(x)$. This time, the promise is that there exists a secret string $s$, where $s$ is not the zero-vector, such that $f(x) = f(y)$ if and only if $x = y \oplus s$. The problem is to find $s$.

To illustrate what we mean, the secret string $s$ is 110 in the following example:

$$\begin{array}{ll} f(000) = 5, & f(100) = 17 \\ f(001) = 4, & f(101) = 42 \\ f(010) = 17, & f(110) = 5 \\ f(011) = 42, & f(111) = 4 \end{array}$$

First, let us examine the query complexity in the classical world. How many queries do we need to solve this problem classically? In classical world, we need on the order of $2^{n/2}$ bits. (To prove that we need $O(2^{n/2})$ queries, pick $s$ randomly and use Yao's Minmax Principle.) By the Birthday Paradox, even with randomness, we still need $\sqrt{2^{n/2}}$ bits.

In quantum computing, we can use Simon's algorithm do this with linear number of circuits, where each circuit is given by:




We apply a Hadamard to the first register, and then query both registers. Then, we measure the second register. After we make the measurement, what is left in the first register is:

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle \longrightarrow \frac{|x\rangle + |y\rangle}{\sqrt{2}}, \text{ where } x \oplus y = s.$$

[3]Simon's original motivation was to try to prove that the quantum model does not provide any exponential speed-up. He failed in his attempt, and what he came up with in the end was a counter-example, which became Simon's algorithm. Shor read this paper and used the general idea to factor integers. So, as we will see later on, Shor's algorithm is Simon's algorithm plus some number theory.

To extract information about $s$, we apply the Hadamard again and see what's left over from the mess.

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

We can't directly extract $s$ from here, but we can extract some useful information about $s$. $(-1)^{x\cdot z} = (-1)^{y\cdot z}$ or else the amplitudes cancel out. This leads us to a linear equation for $s$.

$$\begin{aligned}x\cdot z &= y\cdot z\\ x\cdot z &= (x\oplus s)\cdot z\\ x\cdot z &= x\cdot z \oplus s\cdot z\\ s\cdot z &= 0\end{aligned}$$

We didn't learn $s$, but we did learn that $s$ satisfies the random linear equation $s\cdot z = 0$. We can solve for $s$ using Gaussian Elimination on $n$ linearly independent equations. We only need to run Simon's circuit $O(n)$ times to get $n$ linearly independent equations, so Simon's algorithm runs in $O(n)$ queries. This is an exponential-to-polynominal speed-up!

## 4 $BPP^A \neq BQP^A$

At the end of the day, we didn't prove that $BPP \neq BQP$. However, we did make some progress. Namely, that via some relativization method, we can prove that there exists some $A$, such that $BPP^A \neq BQP^A$.

## 5 Shor's Algorithm [94]

Where Simon's algorithm worked in $Z_2^n$, Shor's algorithm works in $Z_N$. So, $f : [N] \longrightarrow [N]$. The promise is that there exists some $r$, such that $f(x) = f(x+r) = f(x+2r) = ....$ We'll see more on Shor's algorithm and on Hidden Subgroup next lecture!

6.845 Quantum Complexity Theory
Fall 2010