CSE 291 / Math 277A - Quantum Complexity Theory	October 13, 2022
Lecture 7	
Lecturer: Daniel Grier	*Scribe: Dustin Lin*

1 Warm up

Suppose we have an n-qubit quantum circuit construction from $\mathsf{poly}(n)$ Toffoli gates and $\log(n)$ Hadamard gates. What is the smallest classical class that can simulate these circuits?

Observation 1. *Thinking about the Feynman paths/trees from lecture 6, we can imagine a similar tree being constructed based on the circuit's computation. Each Hadamard gate makes the tree split into two branches and the Toffoli gate does not create any split. How many branches of the tree will there be?*

Call this class ξ, we can show that $\xi \subseteq \mathsf{P}$. Since each Toffoli gate can be simulated classically with at most a polynomial slowdown, the only thing we need to take care of is the number of "branches" our computation tree will have. Since we know there will be at most $\log(n)$ Hadamard gates, there will at most be that many amount of splits in our computation. Thus there will be a polynomial amount of splits in the tree ($2^{\log n} = n$), and combined with the fact that the Toffoli gates will keep the tree in poly-depth, searching through such a tree will still be polynomial time.

Observation 2. *This is still true if we relax the definition and allow for any classical gate (not just Toffoli gates) and any single-qubit gate (not just Hadamard). Notice that these transformations will only affect the amplitudes, and not change the size of the tree. The different amplitudes can be kept track of as we traverse the tree.*

2 Quantum Algorithms cont.

In the last lecture we were introduced to Deutsch's algorithm which, from the standpoint of query complexity, allowed us to beat any classical algorithm by a constant number of queries (minimum of 2 queries for classical vs 1 query for quantum). Now we will explore some other promise problems, and see how quantum algorithms compare to classical ones. All of the following problems will be called "promise problems": ones in which there is some guarantee (or "promise") that the input will be of some structure.

2.1 Deutsch-Jozsa problem

Here we are given a function $f\colon \{0,1\}^n \to \{0,1\}$ and want to know is the function constant ($\forall x, y$ we have that $f(x) = f(y)$) or is the function "balanced" (where half of the inputs are evaluated to 0 and the other 1)?

Classically:
Deterministically, we require $2^{n-1}+1$ queries that need to be made to f before we can know for certain. In the worst case all of our previous queries would be the same value until our $\left(2^{n-1}+1\right)$'th query.

Quantum:
Let us describe the algorithm step-by-step, as well as write out mathematically what is going on with our state.

1. Start with the state $|0\rangle^{\otimes n}$ and apply a layer of Hadamard gates:

$$|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$

2. Now let's apply the phase oracle:

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

3. And apply Hadamards again:

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle \xrightarrow{H^{\otimes n}} \frac{1}{2^n} \sum_{x,y\in\{0,1\}^n} (-1)^{f(x)+x\cdot y}|y\rangle$$

Note that $\cdot$ represents the inner product between x and y by viewing them as binary vectors. To see why this is true, we remark that the result of applying an n-fold Hadamard gate on an arbitrary classical state $|x\rangle$ can be written as

$$\begin{aligned} H^{\otimes n}|x\rangle &= \bigotimes_{i=1}^{n} \left(|0\rangle + (-1)^{x_i}|1\rangle\right) = \bigotimes_{i=1}^{n} \left((-1)^{x_i\cdot 0}|0\rangle + (-1)^{x_i\cdot 1}|1\rangle\right) \\ &= \bigotimes_{i=1}^{n} \left(\sum_{y_i\in\{0,1\}} (-1)^{x_i\cdot y_i}|y_i\rangle \right) = \sum_{y\in\{0,1\}^n} (-1)^{\sum_{i=1}^n x_i\cdot y_i} \cdot |y\rangle \,. \end{aligned}$$

Observation 3. *Lets look at the specific case where the state $|y\rangle = |00\ldots0\rangle$ (the all-zeros state). Note that In this case $x \cdot y = 0$, so the above state in step 3 becomes:*

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

Now if the function were constant, then we $f(x) = c$ for some constant $c \in \{0, 1\}$, and:

$$\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{c}|y\rangle = \frac{(-1)^c 2^n}{2^n}|00\ldots0\rangle = \pm|00\ldots0\rangle$$

This requires the amplitude of the all-zeros state to be ±1, or in other words, if we were to measure the state, then we would observe the all-zeros state with probability 1.

Observation 4. *What about the case where f is balanced? Again looking at the all-zeros state from observation 3, notice that the balance for f would result in the amplitudes canceling each other out. This implies that if the function were balanced, then we will measure the all-zeros state with probability 0.*

In summary, we would be able to tell if the function f were constant for balanced by seeing if we observe the all-zeros state. If we do, then it implies the function is constant, if we don't then the function is balanced.

Some additional notes/comments:

- Strictly speaking, if we wanted this to be a BQP algorithm, we can only measure a single bit. To do this, we can simply OR all the bits together and output it to a single register which we measure.
- For a classical model, if we allow the queries to be random, then we only need a constant number of queries to determine whether or not the function is constant or balanced. (The algorithm would be to uniformly draw x and query $f(x)$, and see if we can some queries that are different. We can bound the probability of error by a Chernoff bound.)

Due to the specific problem, it seems like if we allow randomness then asymptotically we aren't doing better with quantum algorithms.

2.2 Berstein-Vazirani problem

Again this will be a promise problem. Here we are given a function $f\colon \{0,1\}^n \to \{0,1\}$ of the form $f(x) = x \cdot s$ (where $s \in \{0,1\}^n$ constant for all x and $\cdot$ is the inner product). Our goal is to find s. The promise here is that the given input function needs to be of that specific form $f(x) = x \cdot s$.

Classically:
Deterministically, we are bounded by n queries. One way to reason about this is that we can first query $x_1 = 100\cdots0$ to find the first bit of s, then $x_2 = 0100\cdots0$ to find the second bit, and so on and so forth to find all n bits of s in n queries. We can not do any better than n queries since if we think about this problem as solving a system of linear equations with n unknowns (bits), then we will at minimum need n equations which correspond to n queries. In this case, randomness does not help us.

Quantum:
Let us describe the algorithm step-by-step, as well as write our mathematically what is going on with our state. The algorithm is **the same** as the one for the Deutsch-Jozsa problem!

1. Start with the state $|0\rangle^{\otimes n}$ and apply a layer of Hadamard gates:

$$|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$

2. Now let's apply the phase oracle:

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

3. And apply Hadamards again:

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle \xrightarrow{H^{\otimes n}} \frac{1}{2^n} \sum_{x,y\in\{0,1\}^n} (-1)^{f(x)+x\cdot y}|y\rangle$$

   (Again, $\cdot$ represents the inner product between x and y.) Note that substituting the function $f(x) = x\cdot s$ gives us: (where the xor comes since we are working with binary vectors appearing as the power of -1)

$$\frac{1}{2^n} \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot s+x\cdot y}|y\rangle = \frac{1}{2^n} \sum_{x,y\in\{0,1\}^n} (-1)^{x(s\oplus y)}|y\rangle$$

Observation 5. *If we look at the case where the state $|y\rangle = s$, then the term $(-1)^{x(s\oplus y)} = 1$ for all x (since the exponent will be 0), and thus the sum over all states would give you a probability of seeing state $|y\rangle = s$ is equal to* 1.

Remark 6. *While this does give us a better separation between what is possible with quantum algorithms (at least asymptotically), this problem isn't technically in* BQP *since this is a* search *problem (looking for a specific s) rather than a* decision *problem where you simply output yes/no.*

2.3 Simon's problem

This problem problem is that you are given a function $f\colon \{0,1\}^n \to \{0,1\}^n$, and the promise that $f(x) = f(y)$ iff $x = y \oplus s$ for some $s \in \{0,1\}^n$ that is constant for the function. Our goal is to find s.

Classically:
We wish to find some collision of x and y such that $f(x) = f(y)$, and be able to find s. Deterministically, we can't do any better than trying an exponential $\Omega(2^n)$ amount of pairs x, y in hopes of finding a collision. If we have access to randomness, then uniformly picking pairs x, y we can expect to find a collision in an

expected $\Theta(\sqrt{2^n})$ queries via the birthday paradox (also called balls and bins bound).

Quantum:
Let us describe the algorithm step-by-step, as well as write out mathematically what is going on with our state. The algorithm is similar but not quite the same as our previous two.

1. Start with the $|0\rangle^{\otimes n}$ and apply a layer of Hadamard gates.

$$|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle.$$

2. Now let's apply the **standard** oracle into an ancilla register:

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle\,|0\rangle \xrightarrow{\text{standard oracle}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|f(x)\rangle$$

3. Now we measure the bits containing the value for $f(x)$. The idea is that after measurement, the states collapse to the superposition of $|x'\rangle, |x\rangle$ that satisfy the $x' = x \oplus s$ for some unknown s. The resulting state would be:

$$\frac{|x\rangle + |x \oplus s\rangle}{\sqrt{2}}$$

4. Applying another Hadamard gives us:

$$\frac{|x\rangle + |x \oplus s\rangle}{\sqrt{2}} \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^{n+1}}} \sum_y \left((-1)^{x\cdot y} + (-1)^{y\cdot(x\oplus s)}\right) |y\rangle$$

5. From here if we measure y in the computational basis, note that the probabilities will be non-zero and add up constructively when the exponents align: $y \cdot x = y \cdot s + x \cdot y$. We can rewrite this as $0 = y \cdot s$.

Observation 7. *Since we are guaranteed to measure some state $|y\rangle$ such that $y \cdot s = 0$, we will need $\mathcal{O}(n)$ additional queries to know what s is. The reason the bound is not exactly n is because (thinking back to solving a system of linear equations to find the secret s using queries) some of our queries to our oracle may result in equations that are linearly dependent.*

Remark 8. *We finally have a problem in which quantum computers are getting an* exponential *advantage over classical computers (n vs $2^{n/2}$ queries). In fact, this problem is the building block and motivation for Shor's algorithm for factoring integers, which can be reduced to a "period finding" algorithm that Simon's problem can be related to.*