COS 533: Advanced Cryptography — Princeton University
Lecture 24 (December 13, 2017)
Lecturer: Mark Zhandry — Scribe: Fermi Ma

Notes for Lecture 24

1 More Quantum Algorithms

1.1 Period Finding

We think of basis states $|x\rangle$ as vectors in $\mathbb{Z}_q^n$. Recall that what the Quantum Fourier Transform (QFT) does the following:

$$QFT|x\rangle = \frac{1}{\sqrt{q^n}} \sum_{y \in \mathbb{Z}_q^n} \omega_q^{\langle x,y\rangle} |y\rangle$$

where $w_q = e^{2\pi i/q}$. Last time, we used this to solve the discrete log problem. Essentially we did period finding. We defined $f(x, y) = g^x h^{-y}$, applied the QFT, and (roughly) the answer popped out. We can also write this function as $f(x, y) = g^{x-ay}$ where $g^a = h$.

Observe that

$$f((x, y) + (a, 1)) = g^{(x+a)-a(y+1)} = g^{x-ay} = f(x, y).$$

This function has the nice property that if I add any multiple of $(a, 1)$ the function stays the same. This is a periodic function, meaning that there's a number or vector such that if I add this number or vector the function stays unchanged.

The quantum algorithm for discrete log recovers this period.

1.2 Shor's Algorithm for Factoring Integers

It turns out that the problem of factoring integers can be rephrased as a period-finding algorithm. Let $N = pq$ (this algorithm works for any N, but this assumption will make things simpler for today).

Step 1) We want to find non-trivial square roots of $1 \mod N$. The trivial square roots are 1 and -1, but there are two more. Recall that the Chinese Remainder Theorem tells us that $\mathbb{Z}_N \cong \mathbb{Z}_p \times \mathbb{Z}_q$ is useful here. Since $1 \mod N$ is also $1 \mod p$ and $1 \mod q$, we know that the square root must be $\pm 1 \mod p$ and $\pm 1 \mod q$. If we pick $+1$ or -1 for both $\mod p$ and $\mod q$, then we get one of the trivial square roots.

However, if we pick $+1 \mod p$ and $-1 \mod q$ or $-1 \mod p$ and $+1 \mod q$, these correspond (via the Chinese Remainder Theorem) to a non-trivial square root of 1. For example, if x is $1 \mod p$ and $-1 \mod q$, we know that $x+1$ is not a multiple of N since it's $2 \mod p$, but it is a multiple of q. So we can take recover q as $gcd(x+1, N)$.

Step 2) We claim we can find non-trivial square roots of 1 with period finding. First we randomly pick $a \leftarrow \mathbb{Z}_N^*$ (this is the multiplicative group of integers mod N which consists of the integers relatively prime to N) We know there exists some r such that $a^r = 1 \mod N$ simply because $\mathbb{Z}_N^*$ is a finite group.

What is $a^{r/2}$? This is a square root of 1 as long as r is even. It turns out that over the random choice of a, there is a constant probability that $a^{r/2}$ is a non-trivial square root of 1.

Consider the function $f(x) = a^x \mod N$. This function is periodic since $f(x+r) = a^{x+r} \mod N$. So we choose a random a, define this function f, and then find the period using the QFT, and then hopefully $a^{r/2}$ is a nontrivial square root of 1.

The problem is that we don't know r. But suppose we had a multiple M of r (it turns out we won't really have a multiple of r either, but we'll deal with that later). Then we can do the following

1. We construct the state
$$|\psi\rangle = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle$$

2. Evaluate f in superposition to get
$$|\psi_2\rangle = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x, f(x)\rangle$$

3. Measure $f(x)$ to get some u. We get
$$|\psi_u\rangle \propto \sum_{x:f(x)=u} |x\rangle = \sum_{t=0}^{\frac{M}{r}-1} |x_0 + tr\rangle$$
where x_0 is the smallest solution to $f(x_0) = u$.

4. Apply the Quantum Fourier Transform. If x_0 happens to be 0, then we get a superposition over multiples of M/r. But even if x_0 isn't 0, we know that this constant shift just leads to a different phase. We won't look at the phase and instead just hide it with a ϕ variable. Ignoring the x_0, we have precisely one

of the examples we worked out in the last lecture (see bottom of page 4 of the lecture 22 notes). Throwing in the extra ϕ phase shift we get

$$QFT|\psi_u\rangle \propto \sum_{s=0}^{r-1} \phi|s\frac{M}{r}\rangle$$

5. Measure to get a multiple of M/r.
6. Repeat this many times with a fixed f and M and then take the GCD of all the multiples of M/r that this procedure produces. With high probability the greatest common divisor will be M/r. From this we can recover r.

Once we have r, we just hope that it's even and that $a^{r/2}$ is in fact a non-trivial square root of 1 (if not, repeat the steps above and try a different r).

What if we don't know M? It turns out we can just guess an M' and as long as its sufficiently large, the QFT gives you something close to the multiples of M/r. We pick $M' \approx O(N^2)$. There are some other tricks you need to play to make this work, but we won't cover this today.

So we know that quantum algorithms are fantastic at finding periods of functions. Note that the quantum algorithm for finding discrete log didn't care what group we were in. Once someone builds a quantum computer they can essentially break discrete log in any group.

1.3 Speeding up NP Problems with Grover's Algorithm

Turns out for general NP problems, we can get a quadratic speed-up, but it won't always give us a polynomial time algorithm (assuming $P \neq NP$).

Consider a boolean function $f : \{0,1\}^n \to \{0,1\}$ for which there exists exactly one x such that $f(x) = 1$. The goal is to find x. We don't know anything else about f, so this is called unstructured search.

Classically, for general f, all we can do is iterate over all possible inputs. Think of f as some oracle that we can query on various points. This takes $O(2^n)$ evaluations of f.

Turns out with quantum we can do $O(2^{n/2})$ which is a quadratic speedup over brute force search.

1. First, we set up the superposition

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$

2. We repeatedly apply a "Grover Iteration", which is a two step process.

   (a) We apply the map that sends $|x\rangle$ to $|x\rangle$ if $f(x) = 0$ and $-|x\rangle$ if $f(x) = 1$. How do we do this? We'll assume we can access the function $f(x)$ in a way that allows us to XOR this function $f(x)$ into the response register $|x, b\rangle \to |x, b \oplus f(x)\rangle$.
   First create the state by appending the polarization at 135 degrees to $|\psi\rangle$ (here $|\psi\rangle$ could either be the original $|\psi\rangle$ or the result of applying this computation a few times) to get
   $$|\psi\rangle(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{2}|1\rangle).$$
   Then we apply f. If we pretend that $|\psi\rangle$ is just $|x\rangle$ for some x, then applying f results in
   $$|x\rangle(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle) \to |x\rangle(\frac{1}{\sqrt{2}}|f(x)\rangle - \frac{1}{\sqrt{2}}|\overline{f(x)}\rangle)$$
   where $\overline{f(x)}$ denotes the complement bit (recall that the output is a single bit).
   If $f(x) = 0$, then we get $|x\rangle(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)$.
   If $f(x) = 1$, we get $-|x\rangle(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)$
   So going back to $|\psi\rangle$, we have that if we apply f to $|\psi\rangle = \sum_x \alpha_x |x\rangle(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)$, we get
   $$\sum_x \alpha_x (-1)^{f(x)} |x\rangle(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle).$$
   We can just throw away the last bit, and we have exactly what we want.

   (b) The second step works as follows. Suppose we start at an equal superposition of all states $|x\rangle$, each with amplitude $\frac{1}{\sqrt{2^n}}$. The first step leaves all the states untouched except for $|x_0\rangle$ where x_0 is the one accepting value for f. The amplitude of $|x_0\rangle$ gets flipped to $-\frac{1}{\sqrt{2^n}}$. The second step operation is merely a reflection about the average amplitude.
   Note that when just $|x_0\rangle$ has amplitude $-\frac{1}{\sqrt{2^n}}$, the average amplitude is just slightly smaller than $\frac{1}{\sqrt{2^n}}$. Rotating about the average sends the amplitude of $|x_0\rangle$ to just below $\frac{3}{\sqrt{2^n}}$, and sends all the other points to something still slightly smaller than $\frac{1}{\sqrt{2^n}}$.
   More precisely, if we have a state $\sum_x \alpha_x |x\rangle$, and define μ to be the average α_x, then this map sends the state to $\sum_x (2\mu - \alpha_x)|x\rangle$.
   We'll explore the implementation details of this in a moment. This transformation is indeed unitary (clearly it's reversible, and it also turns out to be norm-preserving).

Intuition: We can think of step a in the Grover iteration as just flipping the sign of the amplitude of $|x_0\rangle$. Then step b reflects the amplitude of every state $|x\rangle$ about the average amplitude. Every time a Grover iteration occurs, the absolute value of the ampltiude of $|x_0\rangle$ grows while the absolute values of the other amplitudes shrink very slightly. We simply repeat the Grover iterations and then perform a measurement at the end to learn $|x_0\rangle$ with constant probability.

How many iterations do we need to do? After k iterations the amplitude on $x_0 \approx \frac{O(k)}{\sqrt{2^n}}$. If we measure the state now, we get x_0 with probability $O(\frac{k^2}{2^n})$. This is basically where the quadratic speedup comes from. We need to set $k \approx O(\sqrt{2^n})$, and then this probability becomes a constant. We need to work a little harder to prove the errors don't accumulate too fast.

Now we need to implement this "rotating about the average" step of the Grover iteration.

We use QFT_2, the Quantum Fourier Transformation in the $q = 2$ case where $\omega_2 = -1$, and a new gate G. G sends $|0\rangle$ to $|0\rangle$, and $|y\rangle$ to $-|y\rangle$ for all other $y \neq 0$ (this is unitary, but we won't verify it).

First, we apply the QFT_2, then G, then QFT_2 once more. So step b simply maps

$$|\psi\rangle \rightarrow QFT_2 \circ G \circ QFT_2|\psi\rangle$$

It turns out that this exactly implements the reflection about the mean. Roughly speaking, we can think of the G as performing the desired reflection but in a different basis, and the QFT_2 gates are required for the base change.

What if I have more than one accepting input? The problem is that these errors will accumulate too fast, and by the time we're at $O(\sqrt{2^n})$ iterations, we won't get the right answer anymore.

But suppose there are r accepting inputs. The total weight squared on all accepting inputs is approximately $\frac{rk^2}{\sqrt{2^n}}$. Then if we set $k = O(\sqrt{\frac{2^n}{r}})$.

Let's say I don't know how many accepting inputs there are. We're just going to try all the powers of 2 for r.

1.4 Applications of Grover's Algorithm

In cryptography we care about inverting one way functions. We set security parameters like $\lambda = 128$ because we figure that classical computers can't do 2^{128} operations, even if we use the entire world's resources. But since Grover's algorithm brings us down to 2^{64}, suddenly inverting one way functions is within the realm of possiblity.

So inverting one way functions goes from 2^n to $2^{n/2}$ when we go from the classical world to the quantum world. What about finding collisions in collision resistant hash functions?

The birthday paradox says we can do this in $2^{n/2}$ time in the classical world. With Quantum algorithms you can get $2^{n/3}$.

The algorithm that does this uses Grover as a subroutine. It chooses $k = 2^{n/3}$ random inputs $x_1, \ldots, x_k$. We record $z_i = F(x_i)$ for each of these i.

First we verify that there are no collisions in $\{x_i\}$ (if there is one we can stop here, but there probably won't be).

Then we define the function $f(y)$ be 1 if $y \notin \{x_i\}$ and $F(y) \in \{z_1, \ldots, z_k\}$ and 0 else. This function accepts any input that collides with one of the inputs in $x_1, \ldots, x_k$.

Finally, we run Grover's algorithm on f to find our collision. The question is how many iterations do we need to do?

Assume we have a random $f : \{0,1\}^n \to \{0,1\}^n$. The $z's$ form a $\frac{2^{n/3}}{2^n} = \frac{1}{2^{2n/3}}$ of the possible outputs. Each y maps into $\{z_i\}$ with probability $\frac{1}{2^{2n/3}}$. We have roughly 2^n y's. So we have roughly $2^{n/3}$ accepting inputs.

Now we just apply Grover. So the number of iterations we need is $\sqrt{\frac{2^n}{2^{n/3}}} = 2^{n/3}$. It turns out that setting $k = 2^{n/3}$ gives us the optimal balance.