COS 597A: Quantum Cryptography — Princeton University
Lecture 6 (October , 2018)
Lecturer: Mark Zhandry — Scribe: Kevin Liu

Notes for Lecture 6

Today we will see how quantum algorithms can factor integers. This is quite important as factorization is one of the most important assumption in Cryptography. But before switching to factoring, let us look at some applications of Grover's search algorithm.

1 Pre-image search and collision finding

In the last lecture, we have started seeing quantum algorithms. The first one having some 'real' applications is Grover's algorithm. Let's recall the algorithm: assume $F : \{0,1\}^n \rightarrow \{0,1\}$ with δ fraction of inputs accepted. In other words, $\Pr_x[f(x) = 1] = \delta$. Then we can find one accepting input with $O((1/\delta)^{1/2})$ evaluations to f. Classically, what we can do is to just try a random input and with (expected) $O(1/\delta)$ evaluations to find one accepting input. The number of evaluations to f is usually query complexity, the number of queries made to f.

One application of Grover's algorithm is to invert an one-way function. Consider the following problem: given $g : \{0,1\}^n \rightarrow \{0,1\}^m$, given $g(x)$ (but not x) for a random x, the goal is to find any y such that $g(y) = g(x)$. This is called **pre-image search** problem.

A function makes this problem hard is roughly called "one-way function". We can use Grover's algorithm to solve this problem: let $f(y) = 1$ if and only if $g(x) = g(y)$, 0 otherwise; and $\delta \geq 1/2^m$. It gives us an algorithm with only $O(2^{m/2})$ queries. Comparing to a classical algorithm, it is a quadratic speedup.

Another example is **collision finding**: find $x_0 \neq x_1$ such that $g(x_0) = g(x_1)$. It is not harder than preimage search. For simplicity, let us assume g has range size N and is a 2-to-1 function such that for every $g(x)$, it has exactly two pre-images in its domain. We can randomly choose x, compute $g(x)$ and find x' such that $g(x) = g(x')$. Because every image has exactly two pre-images, this algorithm gives a valid pair of collisions with probability 1/2. This problem is indeed easier than pre-image search. Classically, we can use 'birthday algorithm'. The classical algorithm basically does the following: run g on k random inputs, and look for a collision. Here we set $k = N^{1/2}$. Because the probability of having no collision after k queries is $\prod_{i=0}^{k-1}(1 - i/N) = O(e^{-k^2/N})$.

But with quantum computers, one can reduce the query complexity to be $O(N^{1/3})$. The algorithm is an application of Grover's algorithm: We can run g on a bunch of inputs and collect them together as a database $D = \{(x_i, y_i = g(x_i))\}$. And then we set up a search problem (set up a function f) such that $f(x) = 1$ if and only if $g(x)$ collides with one of the values in the database (make sure pre-images do not collide). In other words,

$$f(x) = \begin{cases} 1 & \text{if } g(x) = y_i \text{ and } x \neq x_i \text{ for some } x_i \in D \\ 0 & \text{otherwise} \end{cases}$$

Assuming g is a 2-to-1 function, $\delta = 2k/N$. So the total number of queries is $k + (N/2k)^{1/2}$ (k is the number of queries to prepare the database D and the second term is the queries made by Grover's algorithm) which takes its minimum when $k = O(N^{1/3})$.

	pre-image search	collision finding
classical algorithm	$\Theta(N)$	$\Theta(N^{1/2})$
quantum algorithm	$\Theta(N^{1/2})$	$\Theta(N^{1/3})$

Table 1: query complexity comparison (they are all tight)

Now we are going to see other areas where quantum algorithms really shine.

2 Shor's algorithm: Part 1

2.1 Background

Let us start with factoring with integers. $N = pq$ where p and q are two large primes. Suppose that 3 does not divide $p - 1$ or $q - 1$ (we can also handle the case when it is not true, but for this toy example, let us only deal with 3). It turns out that there exists d such that $x^{3d} = x \pmod{N}$ (in other words, $d^{-1} = 3 \pmod{\phi(N)}$). And moreover if p and q are known, we can efficiently compute d (by extended gcd). In the other hand, if we don't know the factor p and q, you can not compute d (knowing $d \Rightarrow$ computing p and q).

Here is a possible application (of factoring): Assume Alice and Bob are in an insecure channel which an adversary Eve is listening to. Alice wants to share a secret message with Bob without letting Eve knowing it.

Bob chooses random large primes p and q, computes $N = pq$, and $d = 3^{-1} \pmod{\phi(N)}$. And he sends N to Alice. Alice has some message x. She computes $c = x^3 \pmod{N}$ and sends c to Bob. Eve gets both N and c. If we assume computing cube root is hard in this case, then Eve can not recover x. But we know that Bob actually can: $c^d = x^{3d} = x \pmod{N}$. This kind of ideas are roughly known as public key encryption scheme.

But with an algorithm that allows you to factor, the scheme is broken. Given N, Eve can compute p and q which allow him to compute $\phi(N) = (p-1)(q-1)$. By knowing $\phi(N)$, computing $3^{-1} \bmod \phi(N)$ just requires solving the following equation $3x + \phi(N)y = 1$ which is solvable by extended gcd algorithm.

With quantum computers, we can factor N efficiently (taking time polynomial in the number of bits, or $\log N$).

2.2 Quantum algorithm

First, we observe that there are exactly four square roots of 1 mod N. Using Chinse Remainder Theorem, they are $(1,1), (-1,-1) \bmod (p,q), (-1,1), (1,-1)$ (two non trivial roots). Second, finding non trivial roots actually implies factoring. Assume we know $x - 1 = 0 \pmod p$ and $x - 1 = -2 \neq 0 \pmod q$. So $\gcd(x-1, N) = p$.

If we choose random a in Z_N and $\gcd(a, N) = 1$, then there exists an $r > 0$ such that $a^r \bmod N = 1$. Let r be the smallest positive integer and it is called the period of $a \bmod N$. Suppose r is even, then $a^{r/2}$ is a square root of 1 (indeed -1). It can be also shown that for random a, r is even and $a^{r/2}$ is non trivial with reasonable probability.

Our goal now is given a, to compute this r. We rephrase the problem: $f_a(x) = a^x \bmod N$ and we know that there exists an $r > 0$ such that $f_a(r) = 1$, and also we know that $f_a(x + r) = a^{x+r} \bmod N = a^x \bmod N = f_a(x)$ which is a periodic function with period r; we are going to find the period of a periodic function. Recall last week, we have Simon's problem where $f : \{0,1\}^n \to \{0,1\}^m$ given the promise $f(x \oplus r) = f(x)$ and the goal is to find r. They look similar. Here are the idea of the algorithm which is roughly the similar idea for Simon's problem:

1. create the uniform superposition of inputs:

$$\frac{1}{\sqrt{\phi(N)}} \sum_{x=0}^{\phi(N)-1} |x\rangle$$

   (But we dont know what is $\phi(N)$. We will revisit how to do this in the next lecture.)

2. Evaluate f_a on the superposition and get

$$\frac{1}{\sqrt{\phi(N)}} \sum_{x=0}^{\phi(N)-1} |x, f_a(x)\rangle = \frac{1}{\sqrt{\phi(N)}} \sum_{x=0}^{\phi(N)-1} |x, a^x \bmod N\rangle$$

3. Measure $f_a(x)$ registers and we will get y, assume x_0 is the smallest positive

input such that $f_a(x_0) = 1$, then the remaining state is

$$\frac{1}{\sqrt{\phi(N)/r}} \sum_{i=0}^{\phi(N)/r-1} |x_0 + ir, y\rangle$$

4. Apply QFT to all 'x' registers:

   What is QFT_M? takes a state $\sum_{x=0}^{M-1} a_x|x\rangle$, and outputs $\frac{1}{M^{1/2}} \sum_y \sum_x a_x \omega_M^{xy} |y\rangle$ where $\omega_M = e^{2\pi i/M}$. We will see more about QFT in the next lecture.

   Applying QFT, we will get

$$\begin{aligned} \frac{1}{\phi(N)/\sqrt{r}} \sum_y |y\rangle \sum_l w_M^{(x_0+rl)y} &= \frac{1}{\phi(N)/\sqrt{r}} \sum_y |y\rangle w_M^{x_0 y} \sum_l w_M^{ryl} \\ &= \frac{1}{\sqrt{r}} \sum_{y, y=0 \pmod{\phi(N)/r}} \omega_M^{x_0 y} |y\rangle \end{aligned}$$

5. Measure the register and get $\phi(N)/r \cdot k$ for some random k. Collect a bunch of them and apply gcd to recover $\phi(N)/r$.

There are some problems with this algorithm. First is that we do not know $\phi(N)$ so we can not even prepare a superposition of all inputs or recover r. We do know QFT_2 is Hadamard gate. But the second problem is how to implement QFT_M when $M > 2$. We will resolve this two problems in the next lecture.