CS 880: Quantum Information Processing 10/05/10

Lecture 14: Computing Discrete Logarithms

Instructor: Dieter van Melkebeek Scribe: John Gamble

Last class, we discussed how to apply order finding to efficiently factor numbers using a quantum computer. We then outlined how this capability presents problems to certain cryptographic systems, such as RSA. In this lecture, we will discuss a quantum algorithm for period finding of which order-finding is a a special case. We then develop a similar algorithm efficiently compute the discrete logarithm. Finally, we conclude by outlining Diffie-Hellman key exchange, whose security relies on the discrete logarithm being difficult to compute.

1 Period finding

Suppose that we are given some function $f : \mathbb{Z} \to \{0,1\}^m$, and are promised that it is periodic with period $r > 0$. That is, for all x, we know that $f(x+r) = f(x)$. Further, we are guaranteed that no value repeats between periods, so that for all s such that $0 < s < r$, $f(x) \neq f(x+s)$. Our task is to determine the value of r. In order to develop the algorithm, we will first assume that we are also given another integer N such that $r|N$. We will then relax this assumption, so that N need only be an upper bound for r.

1.1 Special case: r divides N

First, we initialize our system in the state

$$|\psi_0\rangle = |0^n\rangle\,|0^m\rangle\,, \tag{1}$$

where $n = \lceil \log N \rceil$, so that N can be represented in the first register. Next, we create a uniform superposition over $\{0, ..., N-1\}$ in the first register. Operating on our our state with the quantum oracle and storing its output in the second register gives us

$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle\,|f(x)\rangle\,. \tag{2}$$

Then, we observe the second register, leaving us in the state

$$|\psi_2\rangle = \frac{1}{\sqrt{N/r}} \sum_{x} |x\rangle\,|z\rangle\,, \tag{3}$$

where z is chosen uniformly at random from $\{f(0), f(1), \ldots, f(r-1)\}$ and the sum runs over all x such that $f(x) = z$. Note that there are N/r such values of x, hence the normalization. We now call the smallest x in our sum x_0 and note that since f has period r, each x can be written as $x = x_0 + jr$ for an integer j. Hence, our state is

$$|\psi_2\rangle = \frac{1}{\sqrt{N/r}} \sum_{j=0}^{N/r-1} |x_0 + jr\rangle\,, \tag{4}$$

where we have suppressed the second register, as we will not need it again. Now, we apply the inverse discrete Fourier transform over $\mathbb{Z}_N$ to the state, which leads to

$$\begin{aligned} |\psi_3\rangle &= F_N^{-1}\,|\psi_2\rangle \frac{1}{\sqrt{N/r}} \sum_{j=0}^{N/r-1} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-2\pi i(x_0+jr)y/N}\,|y\rangle \\ &= \frac{\sqrt{r}}{N} \sum_{y=0}^{N-1} e^{-2\pi i x_0 y/N} \sum_{j=0}^{N/r-1} \left(e^{-2\pi i r y/N}\right)^j |y\rangle\,. \end{aligned} \tag{5}$$

Note that the second summation constitutes a geometric sum, so if ry/N is an integer,

$$\sum_{j=0}^{N-1} e^{-2\pi i(x_0+jr)y/N} = \frac{N}{r}, \tag{6}$$

otherwise it is zero. Thus, we only ever observe a state $|y\rangle$ such that $ry/N \in \mathbb{Z}$. This means that we observe y, a multiple of N/r, distributed uniformly such that $0 \leq y < N$.

Now, we proceed as we did in order finding. We run the algorithm twice, generating two outcomes $y_1 = a(N/r)$ and $y_2 = b(N/r)$. Since a and b are picked uniformly at random, with high probability $\gcd(a,b) = 1$ and so $\gcd(y_1, y_2) = N/r$. Since we were given N, we can compute r and we are done.

Next, we relax the assumption that $r|N$. Intuitively, instead of yielding only random multiples of N/r, the above algorithm will return a distribution of values that are concentrated around integer multiples of N/r, where the concentration will be higher for higher N.

1.2 General analysis

We now analyze the above algorithm for arbitrary r and N, where N is an upper bound for r. We start from the state

$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle\,|f(x)\rangle\,. \tag{7}$$

Next, for analysis purposes we create a modified Fourier transform that makes sense for our periodic function f. Specifically, if we restrict ourselves to the domain $x \in \{0, 1, \ldots, r-1\}$, we can use $|f(x)\rangle$ rather than $|x\rangle$ as our base states. This is because f is not permitted to have any duplicate values each part of the domain. This modified Fourier transform is now defined as

$$\left|\hat{f}(y)\right\rangle = \frac{1}{\sqrt{r}} \sum_{x=0}^{r-1} e^{2\pi i x y/r}\,|f(x)\rangle\,, \tag{8}$$

which is defined for $0 \leq y < r$. By the general properties of the Fourier transform, we can write

$$|f(x)\rangle = \frac{1}{\sqrt{r}} \sum_{y=0}^{r-1} e^{-2\pi i x y/r} \left|\hat{f}(y)\right\rangle, \tag{9}$$

for $0 \le x < r$. However, note that since both f and the amplitudes in (9) are periodic with period r, expression (9) actually holds for any x. Thus, we can write (7) as

$$\begin{aligned}|\psi_1\rangle &= \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle\left(\frac{1}{\sqrt{r}}\sum_{y=0}^{r-1}e^{-2\pi ixy/r}\left|\hat{f}(y)\right\rangle\right)\\ &= \frac{1}{\sqrt{r}}\sum_{y=0}^{r-1}\left(\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}e^{-2\pi ixy/r}|x\rangle\right)\left|\hat{f}(y)\right\rangle \qquad (10)\end{aligned}$$

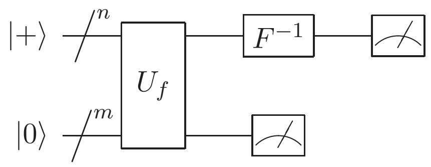
Then, we measure the second register, which gives us a y uniformly at random in $\{0,\dots,r-1\}$. Neglecting the second register, our state is now

$$|\psi_2\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}e^{-2\pi ixy/r}|x\rangle\,. \qquad (11)$$

Next, we apply a standard inverse Fourier transform over $\mathbb{Z}_N$ to this state, in a procedure that we identify as phase estimation with $\omega = y/r$. Hence, measuring the first register gives a state $\left|\widetilde{y/r}\right\rangle$, which is concentrated around $|y/r\rangle$. Now, just as we did for order finding, we can apply the continued fraction expansion to obtain r with finite probability after two independent runs. As anticipated, the only difference between this general treatment and the special case where $r|N$ is that we needed to use phase estimation and a continued fraction expansion.

In order to see that order finding is a special case of period finding, consider the function $f(x) \equiv a^x \mod M$, which has period equal to the order of a. In fact, if we draw the period finding circuit, we can identify an exact correspondence to the eigenvalue estimation and order-finding algorithms:

(a) (12)



Point (a) in the above diagram, corresponding to expression (11), is the same state as (5) in lecture 10, with $\omega = y/r$. Thus, our oracle U_f is equivalent to running the controlled powers of U in figure 2 of lecture 10 (for period-finding) and in the diagram of lecture 12 (for order-finding). Note also that since we do not ever use the result of the lower register, measurement of it is not necessary for the algorithm.

2 The discrete logarithm

Now, we turn our attention to computing the discrete logarithm. Suppose we are given an integer $M > 0$ and a generator g of $\mathbb{Z}_M^*$, the group of all integers mod M that are relatively prime with M under multiplication. Further, suppose that we know the order of $\mathbb{Z}_M^*$, say $R = |\mathbb{Z}_M^*|$ [1]

[1]Note that this is not a further restriction, as R can be computed from the Euler totient function, and it can be shown that the totient can be efficiently computed using the factorization algorithm.

Note that this is a cyclic group in which every element is a power of g. Further, suppose we are given some $a \in Z_M^*$. Then, our goal is to find the smallest integer $l \geq 0$ such that $g^l \equiv a \mod M$. Note that checking that we have a multiple of l is efficient, as it only requires modular exponentiation. However, classically the best known algorithms for finding l are exponential.

In order to construct an efficient quantum algorithm for this problem, consider the function $f : \mathbb{Z}_R \times \mathbb{Z}_R \to \mathbb{Z}_M$, $f(x, y) = a^x g^y \mod M$. Then, we begin with the state

$$|\psi_1\rangle = \frac{1}{R} \sum_{x,y=0}^{R-1} |x\rangle |y\rangle |f(x, y)\rangle . \tag{13}$$

Note that $f(x, y) = a^x g^y = g^{lx+y}$. Now, suppose we observe the third register, resulting in a value g^b, with b chosen uniformly from $\{0, \dots, R-1\}$. Hence, our first two registers are in the state

$$|\psi_2\rangle = \frac{1}{\sqrt{R}} \sum_{(x,y)\in\Lambda} |x\rangle |y\rangle , \tag{14}$$

where $\Lambda = \{(x, y) : 0 \leq x, y < R \text{ and } \ell x + y = b\}$, which is just

$$|\psi_2\rangle = \frac{1}{\sqrt{R}} \sum_{x=0}^{R-1} |x\rangle |b - \ell x\rangle . \tag{15}$$

Next, we apply the inverse Fourier transform over $\mathbb{Z}_R$ to both registers, resulting in

$$|\psi_3\rangle = \frac{1}{R^{3/2}} \sum_{x=0}^{R-1} \sum_{\xi,\eta=0}^{R-1} e^{-2\pi i[\xi x+\eta(b-\ell x)]/R} |\xi\rangle |\eta\rangle . \tag{16}$$

Measuring this state gives us

$$\alpha_{\xi,\eta} = \frac{1}{R^{3/2}} e^{-2\pi i\eta b/R} \sum_{x=0}^{R-1} \left(e^{-2\pi i(\xi-\ell\eta)/R}\right)^x . \tag{17}$$

This is now a geometric sum, so we know that we will only observe states that satisfy $\xi \equiv \ell\eta \mod R$, since otherwise the sum is zero. What this gives us is a uniformly chosen state $|\eta, \ell\xi \mod R\rangle$ for $0 \leq \xi < R$. Hence, if we can compute η^{-1}, we can extract l and we are done. For η^{-1} to exist, we need $\gcd(\eta, R) = 1$, which can be shown to occur with high probability:

$$\Pr[\gcd(\eta, R) = 1] = \frac{|\mathbb{Z}_R^*|}{R} \geq \Omega\left(\frac{1}{\log\log R}\right) . \tag{18}$$

Note that the above procedure can be generalized to other groups. Specifically, the algorithm works for any cyclic group with unique representation such that the group operation is efficiently computable. In the next section, we detail how the computing the discrete logarithm can pose a threat on certain cryptographic systems.

3 Breaking Diffie-Hellman

Consider the Diffie-Hellman protocol, which enables two parties to exchange keys over an insecure channel. The protocol is as follows:

1. The two parties (Alice and Bob) agree publicly on a prime p and a generator g for $\mathbb{Z}_p^*$.
2. Alice picks a randomly from $\mathbb{Z}_p^*$ and sends $A = g^a \mod p$ to Bob.
3. Bob picks b randomly from $\mathbb{Z}_p^*$ and sends $B = g^b \mod p$ to Alice.

Now, the key is $K = g^ab \mod p$. Note that $K = A^b \mod p = B^a \mod p$, so that both Alice and Bob can easily generate K. However, since a and b were never transmitted and were random, it is thought that it is classically difficult to to construct K due to the discrete logarithm being hard to compute.

While this remains an open question, what is clear is that being able to compute the discrete logarithm efficiently trivially breaks the system. The best known classical algorithm for computing the discrete logarithm takes time $2^{O(n)}$ on n-bit inputs, while for factoring it takes time $2^{O(n^{1/3})}$, so classically this protocol seems more robust than RSA. However, this system of key exchange is still vulnerable to an attack by a quantum computer. This also means that certain cryptographic systems, such as El-Gamal encryption, are broken by a quantum computer, as well. Further, many more sophisticated key exchange protocols involving groups over elliptic curves are still vulnerable to a quantum computer, as the quantum algorithm for the discrete logarithm breaks them, too.

In the next lecture, we will discuss the hidden subgroup problem, which is sufficiently general to encompass almost all the applications of efficient quantum algorithms we have seen so far. We will then develop an efficient quantum algorithm to solve it for finitely generated Abelian groups.