Quantum Algorithms 4/18/2023

Lecture 23: Integer Factorization and Discrete Log – DRAFT

Instructor: Dieter van Melkebeek Scribe: Edward Barton, Ivan Hu

Last lecture we instantiated eigenvalue estimation to design an efficient quantum algorithm for order finding. This lecture we discuss the missing piece in the classical part of the algorithm, namely continued fraction expansion. We also discuss an efficient quantum algorithm for the more general problem of period finding. Either can be used as a basis for an efficient quantum algorithm for factoring integers, which is the main topic of this lecture. We end with a discussion of quantum algorithms for computing discrete logarithms.

1 Order finding

Order finding refers to finding the order of some integer modulo some other integer.

1.1 Problem

- **Input:** $a, \mu \in \mathbb{N}$
- **Output:** Smallest positive $r \in \mathbb{N}$ such that $a^r = 1 \bmod \mu$

Note that if a and μ have some factor in common, then $r \in \mathbb{N}$ does not exist. Thus, for r to exist, we require a and μ to be relatively prime, i.e., $\gcd(a, \mu) = 1$. Furthermore, $\gcd(a, \mu) = 1$ implies the existence of such an r that The classical algorithm has a time complexity of $O(\text{poly}(\mu))$. With the quantum algorithm, we will show that this problem can be solved in $O(\text{poly} \log(\mu))$.

1.2 Quantum algorithm

We start with a recap of the algorithm that we developed last lecture. The quantum part consists of eigenvalue estimation applied to the following unitary operator

$$U : |x\rangle \mapsto \begin{cases} |ax \bmod \mu\rangle & \text{for } 0 \leq x < \mu \\ |x\rangle & \text{for } \mu \leq x < N \end{cases}$$

When we use a register for m qubits for the eigenvalue estimates, the process involves computing power of U up to $M \doteq 2^m$, and yields the following: For some unknown $j \in \mathbb{Z}_r$ chosen uniformly at random, a value $y \in \mathbb{Z}_M$ such that $\Pr[|\frac{j}{r} - \frac{y}{M}| \leq \delta] \geq 1 - O(1/(\delta M))$.

The overall algorithm runs the quantum part a number of times, and tries to retrieve r from the samples y obtained. The latter can be done classically. We first note that from a single run, the best we can hope for is retrieving j and r in reduced form as j' and r' where $j' \doteq j/\gcd(j, r)$ and $r' \doteq r/\gcd(j, r)$. By running the process twice independently, yielding j_1 and r_1, and j_2 and r_2, respectively, we showed that

$$\Pr[\text{lcm}(r'_1, r'_2) = r] = \Pr[\gcd(j_1, j_2) = 1] \geq 54\%.$$

Thus, outputting $\mathrm{lcm}(r'_1, r'_2)$ yields an algorithm that produces the correct output with probability more than $1/2$, and this probability can be boosted in the standard way. It remains to show how to efficiently retrieve r' from y and M.

We argued that if $\delta < \frac{1}{2N^2}$ then $\frac{j'}{r'}$ is the unique rational with denominator at most N such that $|\frac{j'}{r'} - \frac{y}{M}|_{\mathbb{T}} \leq \delta$. We now show how in that situation continued fraction expansion allows us to efficiently retrieve r' (and j') from y and M.

1.3 Continued fraction expansion

Continued Fraction Expansion is a technique for rewriting a given $x \in \mathbb{R}$. The steps for the construction are:

- If x is integral, then done.
- Else, write x as $\lfloor x \rfloor + \frac{1}{x'}$ where $x' \doteq \frac{1}{x - \lfloor x \rfloor}$.
- Repeat the process for x'.
- Dropping the remaining term in $\frac{1}{x'}$ in k-th iteration yields the rational $\frac{p_k}{q_k}$ in reduced form, known as the k-th convergent of x.

Example: Consider the case when $x = \pi$:

$$\begin{aligned}
\pi &= 3.14... \\
\pi &= 3 + 0.14... \Rightarrow \frac{p_1}{q_1} = 3 \\
\pi &= 4 + \frac{1}{1/0.14...} \\
\pi &= 3 + \frac{1}{7 + 0.06...} \Rightarrow \frac{p_2}{q_2} = 3 + \frac{1}{7} = \frac{22}{7}
\end{aligned}$$

Recall from primary school that $\frac{22}{7}$ is a very good rational approximation for π, especially compared to its simplicity (small denominator). This showcases an important property of the continued fraction expansion.

Properties:

- $|x - \frac{p_k}{q_k}|$ decreases with k and satisfies $|x - \frac{pk}{qk}| < \frac{1}{q_k^2}$.
- If some fraction $\frac{j}{r}$ satisfies $|x - \frac{j}{r}| < \frac{1}{2r^2}$, then $\frac{j}{r}$ appears in reduced form in the list of convergents.

Rate of Convergence:

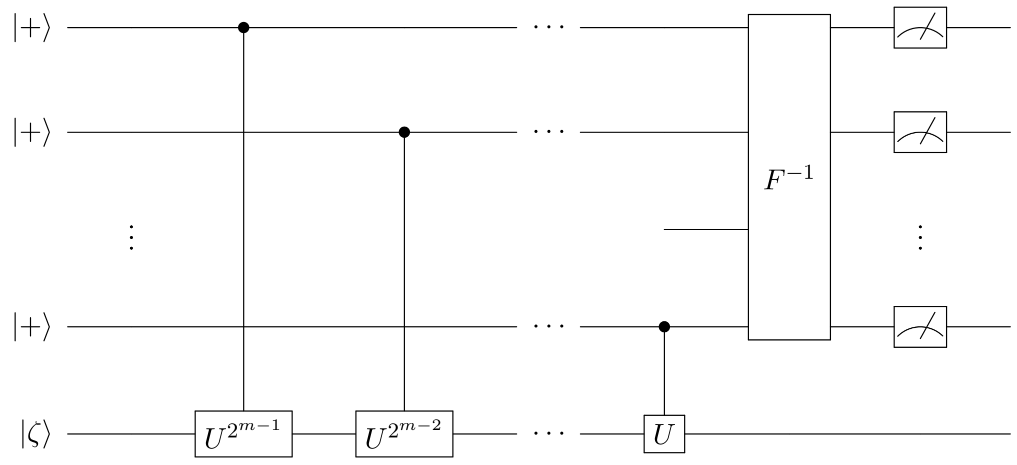
- The process ends iff x is rational.
- $q_{k+1} \geq \max(q_k, 2q_{k-1})$ for $k \geq 2$.
- The list of convergents for rational $x = \frac{y}{M}$ can be computed in time $\mathrm{poly}\log(M)$.

Application to retrieve r': In our case, we can run continued fraction expansion for $x = \frac{y}{M}$ and output the denominator of q_k of the last convergent with $q_k \leq N$ where δ is chosen to be less than $\frac{1}{2N^2}$. By the above properties, if $|\frac{j}{r} - \frac{y}{M}| < \frac{1}{2N^2}$, this returns the correct value of r'.

1.4 Conclusion

We use the following quantum subroutine for eigenvalue estimation, where $|\zeta\rangle = |1\rangle$ and the unitary operator U defined by:

$$U : |x\rangle \mapsto \begin{cases} |ax \bmod \mu\rangle & \text{for } 0 \leq x < \mu \\ |x\rangle & \text{for } \mu \leq x < N \end{cases}$$



The powers of U that are needed can be computed in time poly log(MN) using repeated squaring. The continued fraction expansion in the classical part can be computed in time poly log(N). We need $N \geq \mu - 1$, $\delta < \frac{1}{2N^2}$, and $M \geq \Omega(N^2/\epsilon)$. The resulting algorithm with bounded error ϵ runs in time poly log(μ) for any constant $\epsilon > 0$.

2 Period Finding

The above process of order finding is not how Shor did it. He designed an efficient algorithm for period finding, of which order finding is a special case.

2.1 Problem

- We are given input $f : \mathbb{Z} \to \mathbb{Z}$ for some nonzero integer $p \leq N$ (referred to as the period):

$$f(x_1) = f(x_2) \iff x_1 = x_2 \mod p$$

- Output: p.

2.2 Algorithm

The algorithm uses Fourier sampling, which is to apply the Fourier transform over $\mathbb{Z}_M$ to the first register of $\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle |f(x)\rangle$, and measuring the first register. This results in a $y \in \mathbb{Z}$ such that $|\frac{j}{p} - \frac{y}{m}|_{\mathbb{T}} \leq \delta$ for some $\jmath \in \mathbb{Z}_p$ chosen *almost* uniformly at random, with probability $1 - \epsilon$ provided $M \geq \Omega(1/(\delta\epsilon))$. This can be handled with the same classical analysis as in the previous sections. From this we get an algorithm with bounded error running in time poly $\log(N)$.

3 Integer Factorization

Integer factorization is the problem of taking a positive integer μ and writing it as the product of primes. Note that the complexity of the algorithm is measured in $n = \log \mu$ – the size of the input is n because μ is written in binary. So the trivial algorithm of checking all divisors below μ has a time complexity of $2^{\widetilde{O}(n)}$. Even knowing that μ has a factor less than $\sqrt{\mu}$, given μ is composite, does not help improve the exponent by more than a constant factor.

Currently known classical algorithms such as the general number field sieve do better than that, but they are still exponential, far slower than the quantum algorithm. The heuristic time complexity for the classical algorithm is not proven, but is based on reasonable conjectures on the distribution of primes.

- Classical time complexity: $2^{\widetilde{O}(n^{1/2})}$ (rigorous), $2^{\widetilde{O}(n^{1/3})}$ (heuristic)
- Quantum time complexity: $\widetilde{O}(n^2)$

To solve integer factorization, we can reduce the problem to *splitting*, or writing μ as the product of two smaller integers. By recursively splitting the smaller factors, we can eventually get the prime factorization.

3.1 Splitting to Order Finding

Next, we reduce the problem of splitting to order finding, for which we have a quantum algorithm developed last class. Recall the definition of order finding:

- Input: $a, \mu \in \mathbb{N}$ such that $\gcd(a, \mu) = 1$
- Output: smallest positive $r \in \mathbb{N}$ such that $a^r \equiv 1 \bmod \mu$

Lemma 1. *If* $b^2 \equiv 1 \bmod \mu$ *and* $b \not\equiv \pm 1 \bmod \mu$, *then* $\gcd(b-1, \mu)$ *and* $\gcd(b+1, \mu)$ *are nontrivial factors of* μ.

Proof. We can rewrite the conditions in terms of divisibility: μ divides $b^2 - 1 = (b-1)(b+1)$, but μ divides neither $b - 1$ nor $b + 1$. From this, $b - 1$ and $b + 1$ both contain some, but not all, the factors of μ. □

Lemma 2. *Suppose at least k distinct primes divide μ. For at least $1 - \frac{1}{2^{k-1}}$ of the values of $a \in \mathbb{Z}_\mu^\times$, the order r of a is even, and* $b \equiv a^{r/2} \bmod \mu$ *satisfies* $b \not\equiv \pm 1 \bmod \mu$.

Proof. Omitted; use the Chinese Remainder Theorem. □

This is very useful for finding a factor of μ when μ is composite. If we choose a random element $a \in \mathbb{Z}_\mu^\times$, there is at least a $1/2$ chance that a satisfies the conditions in Lemma 2. With an order finding algorithm, we can compute $b \equiv a^{r/2} \bmod \mu$, and then find the nontrivial factor $\gcd(b+1, \mu)$, allowing us to solve splitting for μ. Repeatedly trying different values of a will greatly increase the chance of success.

Of course, if we somehow choose $a \notin \mathbb{Z}_\mu^\times$, that is fine as well, because $\gcd(a, \mu)$ will be a nontrivial factor of μ (assuming $1 < a < \mu$).

3.2 Full Integer Factorization Algorithm

1. Check whether μ is a prime. If yes, return μ.
2. If $\mu = (\mu')^\ell$ for some integers μ' and $\ell > 1$; then, recursively factor μ' and repeat the factorization ℓ times
3. Pick a random $a \in \{1, \dots, \mu - 1\}$.

   If $\gcd(a, \mu) \neq 1$ then $\mu' := \gcd(a, \mu)$. Else, compute the order r of $a \bmod \mu$.

   If r is even and $b \equiv a^{r/2} \bmod \mu$ satisfies $b \not\equiv \pm 1 \bmod \mu$; then, $\mu' := \gcd(b+1, \mu)$

   Else, try again with a new value of a.
4. Recursively factor μ' and $\frac{\mu}{\mu'}$.

Steps 1 and 2 can be done classically using polynomial time in n. Step 1 can even be done deterministically with the AKS primality test [?]. For step 2, we know that if μ' exists, then $\mu' \geq 2$. Therefore, we need only check $\ell = 2, \dots, \lfloor \log_2 \mu \rfloor$, which can be brute forced.

When we reach step 3, we know that μ must have at least two distinct prime factors, so Lemma 2 guarantees at least a $1/2$ probability of success in step 3. With a quantum order finding algorithm, each iteration of step 3 can be completed using polynomial time in n. The expected number of iterations is 2, and the probability that more than k iterations are required is $1/2^k$, an exponential decrease.

Complexity Polynomial in n (polylogarithmic in μ) with a quantum computer.

Why is this algorithm important? The hardness of factoring integers is used prominently throughout many different cryptography systems. Being able to factor integers quickly breaks many commonly-used schemes, like RSA, which is often used to secure e-commerce transactions. We will discuss RSA next lecture.

4 Discrete Logarithm

For a prime p, we can define the *multiplicative group* $\mathbb{Z}_p^\times = \{x \in \mathbb{Z}_p : \gcd(x, p) = 1\} = \{1, 2, \dots, p - 1\}$, with the group operation of multiplication mod p. In fact, for prime p, this group is a cyclic group – it is isomorphic to $\mathbb{Z}_{p-1}$. Therefore, some elements $g \in \mathbb{Z}_p^\times$ are *generators*. This means that the order of $g \bmod p$ is $p - 1$, the maximum possible order, and the set $\{g, g^2, \dots, g^{p-1}\}$ is a permutation of the elements of $\mathbb{Z}_p^\times$.

In the discrete log problem, we are given a prime p and a generator $g \in \mathbb{Z}_p^\times$ as parameters, and the input is some $a \in \mathbb{Z}_p^\times$. The output is the exponent l such that $g^l \equiv a \bmod p$.

Similar to integer factorization, the discrete log problem has no known efficient classical algorithm, but there is an efficient quantum algorithm based on the hidden subgroup problem for abelian groups covered in lecture 20. The respective time complexities are shown below, where $n = \log p$.

- Classical time complexity: $2^{\widetilde{O}(n^{1/2})}$ (rigorous), $2^{\widetilde{O}(n^{1/3})}$ (heuristic)
- Quantum time complexity: $\widetilde{O}(n^2)$

The presumed hardness of the discrete log problem was used as the basis for cryptographic systems like the Diffie-Helman key exchange protocol, which we will discuss next lecture.