Lecture 11: Order finding (continued); reducing factoring to order finding

March 2, 2006

Solving order finding using phase estimation (continued)

We will begin this lecture by finishing the discussion of how the order finding problem can be solved via phase estimation.

Assume that we are given $a \in \mathbb{Z}_N^*$, and our goal is to calculate the order of a in $\mathbb{Z}_N^*$, which we suppose is r. Recall that we defined a unitary operation M_a so that

$$M_a \left|x\right\rangle = \left|ax\right\rangle$$

for every $x \in \mathbb{Z}$. Then for

$$\left|\psi_k\right\rangle = \frac{1}{\sqrt{r}}\left(\left|1\right\rangle + \omega^{-k}\left|a\right\rangle + \omega^{-2k}\left|a^2\right\rangle + \cdots + \omega^{-k(r-1)}\left|a^{r-1}\right\rangle\right)$$

we have $M_a \left|\psi_k\right\rangle = \omega^k \left|\psi_k\right\rangle$.

Last time we considered what happens when we perform phase estimation on the state $\left|\psi_1\right\rangle$. The eigenvalue associated with $\left|\psi_1\right\rangle$ is $\omega = e^{2\pi i(1/r)}$. Assuming we were to perform the phase estimation procedure several times and take the most commonly appearing result, with very high probability we will have an approximation $j/2^m$ that is close to $1/r$. Our guess for r would therefore be $2^m/j$, rounded off to the nearest integer. We determined that if $m = 2n$ and the phase estimation procedure succeeds in finding a best m bit approximation $j/2^m$ to $1/r$, then rounding $2^m/j$ to the nearest integer will indeed give r.

Unfortunately we do not know how to get our hands on $\left|\psi_1\right\rangle$. In fact it seems like one would need to know r before preparing it, but if we know r there is no need to bother. There is no known way to prepare $\left|\psi_1\right\rangle$ without knowing r, so a different approach is required. Instead, we considered running the phase estimation procedure on the state

$$\left|1\right\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\left|\psi_k\right\rangle .$$

We argued that this would give a result that is indistinguishable from running the phase estimation procedure on an eigenvector $\left|\psi_k\right\rangle$ for a random value $k \in \{0, \ldots, r-1\}$ chosen uniformly. In other words, the phase estimation procedure would give an approximation

$$\frac{j}{2^m} \approx \frac{k}{r}$$

for $k \in \{0, \ldots, r-1\}$ chosen uniformly. The question we must now address is: if you have enough precision, can you now find r from such an approximation? It is important for us to keep in mind that k will not be known explicitly when we attempt to find r.

The answer is that you cannot be guaranteed to find r given just one sample, but if you repeat the process several times (each time for a different k) you can find r with high probability. The way to do it is to use the *continued fraction algorithm*. I described how this algorithm works in lecture but I will not include a description in these notes. If you are interested in learning more about the algorithm, I would be happy to give you some good references. We can sum up what the continued fraction algorithm accomplishes in the following fact:

Fact. Given a real number $\alpha \in (0,1)$ and $N \geq 2$, there is at most one fraction

$$\frac{x}{y}$$

with $0 \leq x, y < N$, $y \neq 0$, $\gcd(x,y) = 1$ and

$$\left|\alpha - \frac{x}{y}\right| < \frac{1}{2N^2}.$$

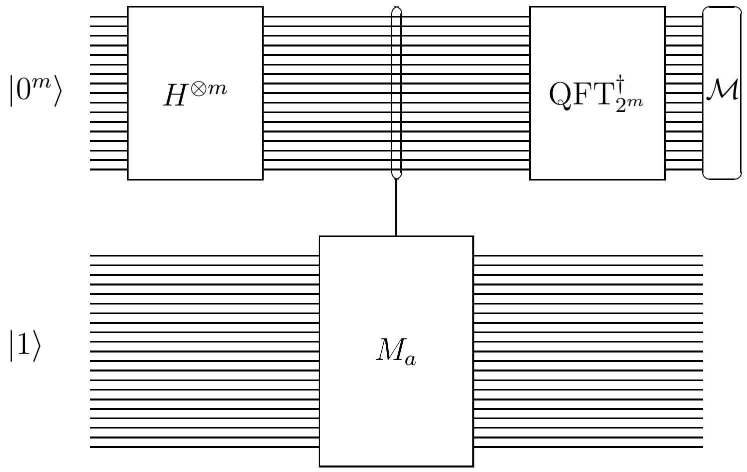
Given α and N, the continued fraction will find x and y (if they exist) in $O((\lg N)^3)$ bit operations.

So, by again setting $m = 2n$, applying phase estimation, and then applying the continued fraction algorithm to the result, we can find x and y such that

$$\frac{x}{y} = \frac{k}{r}.$$

In other words, we can find k/r **in lowest terms**. This may fail to find r if $\gcd(k,r) > 1$, but we know that at least y divides r. Repeating several times (each time for a different k) and taking the least common multiple of the resulting y values gives r with high probability.

So, in the end, the algorithm is as follows. First, choose $m = 2n$ and run the phase estimation procedure:



(It can be run sequentially several times to improve accuracy.) Plug the measurement outcome $j/2^m$ along with N into the continued fraction algorithm, which gives a fraction x/y. Repeat the entire process several times, taking the least common multiple of the y values. The result will be r with high probability. The entire process is known as Shor's algorithm for order finding.

Reduction of factoring to order finding

I've mentioned several times that we can factor integers efficiently using quantum computers, and that this is the problem solved by Shor's algorithm. Really, it turns out that "Shor's algorithm" is more of a technique than a specific algorithm, and it can be used to solve an interesting collection of problems. (Order finding is one of the simplest to describe.)

Now let us see that the integer factoring problem can be efficiently solved given an algorithm for order-finding. In other words, factoring *reduces* to order-finding. To be specific, it is a randomized polynomial-time Turing reduction, which is probably different from (but similar to) the types of reductions you might have seen in CPSC 313 and 413. The idea, however, is essentially the same—if you have an efficient algorithm for order finding, then the reduction gives you an efficient algorithm for factoring.

Integer Factorization

Input:	A positive integer $N \geq 2$.
Output:	A prime factorization $N = p_1^{k_1} \cdots p_m^{k_m}$.

Let us note a few facts, which we will not prove or discuss in any detail. First, if N is a prime number or a prime power (i.e., p^k for some prime p and integer $k \geq 1$), then there are efficient classical algorithms for solving the integer factorization problem in these cases. So, we can imagine that we first run such an algorithm on the input N; if it succeeds then we are done, otherwise we continue on under the assumption that N has at least two distinct prime factors (i.e., $m \geq 2$).

Next, it is enough to have an algorithm that takes a composite N as input and just finds two integers $u, v \geq 2$ such that $N = uv$. If we have such an algorithm, we can run it recursively (interleaved with the classical algorithm for prime powers) to find a complete prime factorization of N.

Finally, if our goal is to find integers $u, v \geq 2$ such that $N = uv$ for N an **even** composite number, then we really don't need to work very hard; let $u = 2$ and $v = N/2$.

Now, consider the process described in Figure 1. Why does it work? The basic idea is as follows. Suppose that the random choice of a is in $\mathbb{Z}_N^*$ (which is very likely), and that the order r of a is even. Then

$$a^r \equiv 1 \pmod{N}$$

so

$$N \mid a^r - 1.$$

Because $a^r - 1 = (a^{r/2} + 1)(a^{r/2} - 1)$ we have

$$N \mid (a^{r/2} + 1)(a^{r/2} - 1).$$

It cannot happen that $N \mid (a^{r/2} - 1)$, because this would mean that r was not the order of a after all. If we are lucky and it is not the case that $N \mid (a^{r/2} + 1)$, then we know the algorithm will work. This is because the factors of N are then necessarily split between $(a^{r/2} + 1)$ and $(a^{r/2} - 1)$, so computing $\gcd(a^{r/2} - 1, N)$ will reveal a nontrivial factor of N.

```
Input: an odd, composite integer $N$ that is not a prime power.
Repeat
    Randomly choose $a \in \{2, \ldots, N-1\}$.
    Compute $d = \gcd(a, N)$.
    If $d \geq 2$ then                              /* We've been incredibly lucky. */
        Return $u = d$ and $v = N/d$.
    Else                                            /* Now we know $a \in \mathbb{Z}_N^*$. */
        Let $r$ be the order of $a$ in $\mathbb{Z}_N^*$.       /* Requires the order finding algorithm. */
        If $r$ is even then
            Compute $x = a^{r/2} - 1 \pmod{N}$.
            Compute $d = \gcd(x, N)$.
            If $d \geq 2$ then
                Return $u = d$ and $v = N/d$.       /* Answer is found. */
Until answer is found (or you get tired).
```

Figure 1: Reduction of factoring to order finding

Each iteration of the loop therefore fails to give an answer if either r is odd or r is even but N divides $a^{r/2} + 1$. The probability that neither of these events occur is at least $1/2$. (In fact, it is at least $1 - 2^{m-1}$ for m the number of distinct primes dividing N. This is why the assumption that N is not a prime power was important. Also, the analysis of this fact requires that N is odd.)