L. Hales
Vazirani Class #8
October 2, 1997

Shor's Factoring Algorithm

1 Introduction

The factoring problem can be posed in the following manner:

GIVEN: A composite N.

FIND: An N_1 such that $N_1 \neq 1, N$ and $N_1 | N$.

This problem is widely assumed to be hard, even for the restricted case when N is the product of 2 primes. Moreover, this problem forms the basis for the well-known RSA public-key cryptosystem, hence finding an efficient factoring algorithm is of great practical interest. There do exist subexponential ($2^{\sqrt[3]{logN}}$) randomized algorithms for factoring. Today we will present Shor's quantum algorithm for factoring. We begin with some number theoretic preliminaries. Throughout our discussion we will assume that N is odd.

2 Number Theoretic Preliminaries

It is well known that factoring can be reduced to the following problem:

GIVEN: A composite N.

FIND: An x such that $x^2 \equiv 1 (mod N)$ and $x \not\equiv \pm 1 (mod N)$.

Suppose we have found x as above. Then $x^2 - 1 = (x-1)(x+1) = kN$ for some k. Since $N | (x-1)(x+1)$ while $N \not| (x+1)$ and $N \not| (x-1)$, some nontrivial factor of N must divide $x+1$. To find this factor it suffices to find $(N, x+1) =$ the greatest common divisor of N and $x+1$, which can be done via Euclid's algorithm in $O(n^3)$ time.

Example: Suppose that $N = 15$. Then $4^2 \equiv 1 (mod N)$ while $4 \not\equiv \pm 1 (mod N)$ hence $(15, 4-1) = 5$ and $(15, 4+1) = 3$ are both nontrivial factors of 15. ∎

To factor N, then, it suffices to find a nontrivial square root of 1 in Z_N^*, where Z_N^* is the group whose underlying set is $\{x | (x, N) = 1, 1 \leq x \leq N\}$ and whose group operation is $mod N$ multiplication.

Definition 1 *The order of an element a in a group G, which we shall denote $ord_G(a)$ (or $ord(a)$ when the group G is clear from the context), is the least integer r such that $a^r = 1_G$.*

Definition 2 $\Phi(n) = |\{x | 1 \leq x \leq n \text{ and } (x, n) = 1\}|$. *($\Phi(n)$ is commonly referred to as the Euler phi function.)*

The following claim implies that if we can compute the order of $a \in Z_N^*$ then we can find a nontrivial square root of one in Z_N^* (and therefore factor N) with high probability.

Claim 3 $\Pr_{a \in_R Z_N^*}[r = ord(a) \text{ is even and } a^{r/2} \not\equiv \pm 1] \geq 1/4$.

Proof: Let $p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$ be the prime factorization of N. By the Chinese remainder theorem, $Z_N^* \cong Z_{p_1^{e_1}}^* \times Z_{p_2^{e_2}}^* \times \ldots \times Z_{p_n^{e_n}}^*$. Let $\Phi(N) = 2^l m$ and $\Phi(p_i^{e_i}) = 2^{l_i} m_i$ where m and the m_i are odd. By the following

Fact 4 *$Z_{P_i^{e_i}}^*$ is cyclic.*

we can fix generators $g_1, \ldots, g_n$ of $Z_{p_1^{e_1}}^*, \ldots, Z_{p_n^{e_n}}^*$ respectively. Then choosing $a \in_R Z_N^*$ is equivalent to choosing $x_i \in_R \{1, 2, ...2^{l_i} m_i\}$ independently for $1 \leq i \leq n$ (by letting $a = (g_1^{x_1}, g_2^{x_2}, \ldots, g_n^{x_n}) \in Z_{p_1^{e_1}}^* \times Z_{p_2^{e_2}}^* \times \ldots \times Z_{p_n^{e_n}}^*$).

The following two lemmas together yield our claim:

Lemma 5 $\Pr_{a \in_R Z_N^*}[r = ord(a)$ *is even* $] \geq 1/2$.

Proof: The order of a in Z_N^* is the LCM of the set $\{\frac{2^{l_1} m_1}{x_1}, \frac{2^{l_2} m_2}{x_2}, \ldots, \frac{2^{l_n} m_n}{x_n}\}$. Notice that since N is odd, each p_i is an odd prime and thus $\Phi(p_i^{e_i})$ is even and $l_i > 0$. Thus if any of the x_n is odd then this LCM must be even. Since each x_n is chosen at random this probability is at least $\frac{1}{2}$. ■

Lemma 6 $\Pr_{a \in_R Z_N}[a^{r/2} \not\equiv \pm 1 | r = ord(a)$ *is even* $] \geq \frac{1}{2}$.

Proof: Fix a and $r = ord(a)$. $a^{r/2}$ is the element $\prod g_i^{x_i r/2}$. Note that there are only two square roots of 1, namely ± 1, in each $Z_{p_i^{e_i}}^*$ (this follows easily from $Z_{p_i^{e_i}}^*$ cyclic), and thus the only square roots of 1 in Z_N^* are of the form $(\pm 1, \pm 1, \ldots, \pm 1)$, with $1 = (1, 1, ...1)$ and $-1 = (-1, -1, ... -1)$.

We know that the $g_i^{x_i r/2}$ are not identically 1 since then r would not be the order of a. Thus we need merely to bound the probability that the $g_i^{x_i r/2}$ are all -1. The only way this can happen is if for each i the highest power of 2 dividing $x_i r$ is l_i. Suppose we have chosen x_1. Let k be the highest power of 2 dividing x_1. In order for $g_1^{x_1 r/2}$ to be -1 the highest power of two dividing r must be $l_1 - k > 0$. The probability of choosing x_2 so that the highest power of 2 dividing it is exactly $l_2 - (l_1 - k)$(and thus $g_2^{x_2 r/2} = -1$) is less than or equal to 1/2, as desired. ■ ■

3 Simplified Quantum Algorithm for determining the order of $a \in Z_N^*$

Suppose we are given N and a random $modN$ residue a. We wish to find $r = ord_{Z_N^*}(a)$. Suppose further that we have been given an integer q such that $r|q$. (Our quantum algorithm will involve a Fourier transform over Z_q. The assumption that $r|q$ allows us to see the idea of the algorithm clearly-later we will show how to drop this assumption in exchange for a more complicated algorithm).

Algorithm 7 *Our initial superposition is:*

$$|N>|a>|0 mod q>|0 mod N>$$

Performing a fourier transform over Z_q we obtain:

$$\frac{1}{\sqrt{q}}|N>|a>\sum_{x\in Z_q}|x mod q>|0 mod N>$$

By a classical computation we get:

$$\frac{1}{\sqrt{q}}|N>|a>\sum_{x\in Z_q}|x mod q>|a^x mod N>$$

Now we measure the value $a^x modN$ without disturbing the other bits on the tape. Let a^k be the observed value of $a^x modN$. Our superposition will then collapse to the following:

$$\frac{1}{\sqrt{q/r}}|N>|a>|a^k mod N>\sum_{x\in Z_q, x=lr+k}|x mod q>$$

In other words, we obtain a uniform superposition over some coset of $<r>$ (the subgroup of Z_q generated by r). Applying another fourier transform over Z_q we will then obtain a uniform superposition over $\frac{Z_q}{<r>}$:

$$\frac{1}{\sqrt{r}}|N>|a>|a^k mod N>\sum_{l=1 to r}|\frac{lq}{r}>$$

When we observe at this point we will see lq/r for a random l between 1 and r. With significant probability $(l,q)=1$ in which case $(q,\frac{lq}{r})=\frac{q}{r}$.

We can compute $(q,\frac{lq}{r})$, divide q by it, then check to see if this is truly the order of a. If so we are done, if not we can repeat the algorithm.