CS 681: Computational Number Theory and Algebra **Lecture 14**

A Polynomial time algorithm for Primality Testing

Lecturer: Manindra Agrawal **Scribe: Chandan Saha**

Septembor 9, 2005

In the previous lecture we have proved the two size reduction lemma. It follows that:

If

1. $T = \{X^j + a \mid 0 \le j < r, 0 \le a \le 2\sqrt{r}lgn\}$
2. $p > t > 4log^2 n$
3. ψ is linear on T

then $n = p^j$ *for some* $j \in N$.

1 The Algorithm and its correctness

1.1 Algorithm

Input: integer $n > 1$.

1. Test if $n = m^j$ for some $j > 1$. If yes output COMPOSITE.
2. Find the smallest r such that $order_r(n) > 4log^2 n$.
3. If $1 < (a, n) < n$ for some $a \le r$, output COMPOSITE.
4. If $n < r$, output PRIME.
5. For $1 \le a \le 2\sqrt{r}logn$ do
   if $((X + a)^n \ne X^n + a(\bmod X^r - 1, n))$, output COMPOSITE.
6. output PRIME.

1.2 Correctness

Theorem 1.1 *The algorithm above returns PRIME if and only if n is prime.*

Lemma 1.1 *If n is PRIME, the algorithm returns PRIME.*

Proof: If n is prime then either the algorithm outputs PRIME in Step 4 or else the condition tested in Step 5 never holds and the algorithm returns PRIME in Step 6. ∎

Lemma 1.2 *If the algorithm returns PRIME then n is prime.*

Proof: If the algorithm returns PRIME in Step 4 then n is indeed prime. For the rest of the proof, consider that the algorithm returns PRIME in Step 6. This implies that,

$$\begin{aligned}\psi(X+a) &= (X+a)^n \ (\bmod n, X^r-1)\\ &= X^n + a \ (\bmod n, X^r-1)\\ &= \psi(X) + a \ (\bmod n, X^r-1)\end{aligned}$$

for $0 \leq a \leq \ 2\sqrt{r}logn$. Replacing X by X^j we get,

$$\begin{aligned}\psi(X^j+a) &= \psi(X^j) + a \ (\bmod n, X^{jr}-1)\\ &= \psi(X^j) + a \ (\bmod n, X^r-1)\\ &= \psi(X^j) + a \ (\bmod p, h(X))\end{aligned}$$

By definition, $G = \{\phi^i\psi^j(X) \mid i, j \geq 0, X \in F\} = \{X^{n^j p^i}\}$. Choose the irreducible factor $h(x)$ of the polynomial $x^r - 1$ in $F_p[x]$ that has an r^{th} primitive root of unity over the field F_p (this can always be done). This choice of $h(x)$ makes $t = \mid \{n^i p^j(r) \mid i, j \geq 0\} \mid$. This implies that $t \geq order_r(n) > 4log^2n$. Also we have $r \geq t$ and $p > r$ (from Step 3). Therefore, $n = p^j$ for some j. Since at Step 6 we have that $n \neq m^j$ for any m and any $j > 1$, we get $n = p$. ∎

2 Time Complexity Analysis

We will need the following fact about the lcm of the first m numbers.

Lemma 2.1 *Let LCM(m) denotes the lcm of the first m numbers. For $m \geq 7$ we have $LCM(m) \geq 2^m$.*

The following lemma bounds the magnitude of r.

Lemma 2.2 *There exists an $r \leq 16log^5n$ such that $order_r(n) > 4log^2n$.*

Proof: Consider the product

$$A = n \cdot \prod_{j=1}^{4log^2n} (n^j - 1)$$

Say an r is *bad* if either $r \mid n$ or $order_r(n) \leq 4log^2n$. It is easy to see that all *bad* $r's$ divide A. Moreover,

$$A < n \cdot n^{\sum_1^{4log^2n} j} \leq 2^{16log^5n}$$

Therefore by Lemma 2.1 there exists an $r \leq 16log^5n$ such that r does not divide A, implying that r is not *bad*. If now $(r,n) = 1$ then we are done. If $(r,n) > 1$ then $s = \frac{r}{(r,n)}$ does not divide A and s is relatively prime to n. This implies that $order_s(n) > 4log^2n$. ∎

Theorem 2.1 *The asymptotic time complexity of the algorithm is* $\tilde{O}(log^{21/2}n)$.

Proof: Time taken in Step 1 is $\tilde{O}(log^3n)$. In Step 2, time spent to check if $order_r(n) > 4log^2n$ is $\tilde{O}(log^2n)$ for any r ($logr$ factor hidden). Therefore, Step 2 takes $\tilde{O}(rlog^2n)$ total time. Execution of Step 3 can be done in $\tilde{O}(rlogn)$ time. Time taken to compute $(X+a)^n$ and X^n in the ring $Z_n[X]/(X^r - 1)$ is $\tilde{O}(rlog^2n)$ (hiding the $logr$ factor) for any fixed a. Therefore, total time spent in Step 4 is $\tilde{O}(r^{\frac{3}{2}}log^3n)$. The theorem follows from Lemma 2.2. ∎