CS 538 Computational Complexity	**Scribed by Aynur Dayanik**
Prof. Eric Allender	**March 30, 1998**

In this lecture, we will prove that Factoring is in UP $\cap$ CoUp. First, we will show that Primes is in NP and in UP where the language Primes defined as follows:

$$Primes \;=\; \{x \text{ written in binary} :\; x \text{ is a prime number}\}$$

We need some facts from number theory.

Fact: p is a prime number $\Longrightarrow$ $\forall g \in \{1, \cdots, p-1\}$ $g^{p-1} \equiv 1 \pmod{p}$.

Definition: The order of $g \pmod{p}$ is the least i such that $g^i \equiv 1 \pmod{p}$. This always divides the size of $Z_p{}^*$ which is the multiplicative group mod p.

If c is a divisor of $q-1$ and $g^c \not\equiv 1 \pmod{q}$, then the order of g is **not** c.

Fact: p is prime $\Longleftrightarrow \exists g \in \{1, \cdots p-1\}$ such that $g \pmod{p}$ has order $p-1$.

To show Primes $\in$ NP, we need to have a short "proof" of primality. We need a table as follows:

p
prime divisors of $p-1$: $q_1,\ q_2,\ \cdots\ q_k$
prime divisors of $q_1 - 1,\ q_2 - 1,\ \cdots\ q_k - 1$
$\cdots$
$\cdots$
$\cdots$

In this table, we keep the binary representation of the number p, the prime divisors of $p-1$, and so on. Therefore, each row contains the binary representation of the prime divisors of the one less of the prime numbers kept in the upper row. The length of the binary representation of p is an upper bound on the number of rows in this table.

For each "prime" q in the table, where the prime factors of $q-1$ are $r_1, r_2, \cdots r_n$, consider the following algorithm:

```
guess a $g \in \{1, \cdots q-1\}$
verify that $g^{q-1} \equiv 1 (\bmod q)$
    and $\forall i g^{q-1/r_i} \not\equiv 1 \pmod{q}$
    i.e. see if the order of $g \bmod q$ is $q-1$
```

The above algorithm provides a proof that q is prime if $r_1, r_2, \cdots r_n$ are primes. Inductively, we can see that the items in the bottom row are primes, and this implies that the items on the next row are prime.

Thus, Primes $\in$ NP. Now let us show that Primes $\in$ UP. For our UP algorithm, we will guess this same table. The NP algorithm that guesses such a table and checks its correctness has at most one accepting path. Therefore, there is exactly one such table that is correct. Hence, Primes $\in$ UP.

Fact: { tables like this } $\in$ P.

Numbers are "composed" of prime factors. If these components are large, then the number is "chunky". Otherwise, they are "smooth". (Smooth is a technical term in number theory.) There are many smooth numbers.

Fact: $|\{j < p : \text{all of } j\text{'s prime factors are} \leq \log^3 p\}| \geq \sqrt{p}$.

Fact: If p is prime, then $x^h - 1$ has $\leq h$ roots mod p.

The last two facts are from number theory and we don't prove them here.

Lemma: Given p and a list $q_1, q_2, \cdots, q_k$ of the prime factors of $p-1$, we can determine in polynomial time if p is prime.

Proof: Let's look at the algorithm.

```
For all $j \leq \log^3 p$,
    check if $j^{p-1} \equiv 1 \pmod{p}$
        if not, $p$ is COMPOSITE
end for
compute $h_j$ (order of $j \mod p$)
we can compute it by the following loop:
    $h_j := p - 1$
    repeat the following:
        check that $gcd(j^{h_j} - 1, p) = p$
        if not, $p$ is composite
        else, pick a divisor $q_i$ and look at $m = gcd(j^{h_j/q_i} - 1, p)$
        if $m = p$ then
            $h_j := h_j/q_i$
        if $m \notin \{1, p\}$, then COMPOSITE
    until for all $i$, $gcd(j^{h_j/q_i} - 1, p) = 1$.
let $h$ = the least common multiple $(h_1, h_2, \cdots, h_{\log^3 p})$
if $h \geq \sqrt{p}$, then $p$ is PRIME
else $p$ is COMPOSITE
```

Let's prove why the above cases are true:

- $h \geq \sqrt{p} \Longrightarrow p$ is prime.

  Let l be any prime divisor of p. Then $j^{h_j} \equiv 1 (\text{mod } p) \Longrightarrow j^{h_j} \equiv 1 (\text{mod } l)$. If h_j is not the order mod$l$, then $j^{h_j/q_i} \equiv 1 (\text{mod } l)$. So, $gcd(j^{h_j/q_i} - 1, p) = l$. Thus, h_j is the order of $j \mod l$. (If l was this gcd, then we would have already announced that p is composite.) Then, h_j divides $l - 1$.

  Thus, if $h \geq \sqrt{p}$, then *any* prime l dividing p must be greater than h, and this implies $l \geq \sqrt{p}$, so p is prime.

- $h \leq \sqrt{p} \Longrightarrow p$ is composite.

  $j^h \equiv 1 \pmod{p}$ for all $j \leq \log^3 p$.

  If c is smooth, then $c^h \equiv 1 \pmod{p}$. Because c is the product of a bunch of numbers $q_1, q_2 \cdots q_k$ all of which are small, and for any q_i, $q_i^h \equiv 1 (\text{mod } p)$. Thus, $c^h = q_1^h, q_2^h \cdots q_k^h \equiv 1 (\text{mod } p)$.

  Thus, all smooth numbers are roots of $x^h - 1$. There are $\geq \sqrt{p} > h$ smooth numbers, so $x^h - 1$ has $> h$ roots. This implies p is composite.

Definition:

$$Factoring = \{(x, i, b) \mid i^{th} \text{ bit of the prime factorization of } x \text{ is } b\}$$

where the prime factorization is written as $(p_1, e_1, p_2, e_2, \cdots, p_r, e_r)$. Namely $x = {p_1}^{e_1} {p_2}^{e_2} \cdots {p_r}^{e_r}$ where $\forall i \in \{1, \cdots, r\}, p_i < p_{i+1}$ and $e_i > 0$. So, this prime factorization is unique.

Corollary: Factoring $\in$ UP $\cap$ CoUP.
Proof: To see why Factoring $\in$ UP, consider the following nondeterministic Turing machine U:

```
on input $(x, i, b)$
    $\exists$ guess a prime factorization of $x$
    check if its $i^{th}$ bit is $b$
        if yes, accept
        else, reject
```

Since the prime factorization of x is unique, the "yes" answer can happen at most once. It is clear that this machine U accepts Factoring and is unambiguous. Hence, Factoring $\in$ UP.

Now, it is easy to show that Factoring is in CoUP. Our CoUP machine will be the same as our UP machine U, except that it will check whether the i^{th} bit of the prime factorization of x is not b on the input (x, i, b). Hence, Factoring $\in$ CoUP.

In the next lectures, we will discuss probabilistic complexity classes. Today, we will define RP which is a probabilistic complexity class defined as follows:

RP $= \{A : \exists$ an NP machine M accepting A such that $x \in A$, $Prob(M(x)$ accepts$) \geq 1/2\}$.

Fact: Primes $\in$ CoRP. (Fairly easy proof. See Solovey-Strassen test in *Primality and Cryptography* by Evangelos Kranakis, 1986.
Fact: Primes $\in$ RP. (Difficult proof, see Adleman and Huang (STOC '87).)

Note that although efficient probabilistic algorithms for Primes are known, no such algorithm is known (or expected) for Factoring.

In the next lecture, we'll show that SAT $\in P/poly \Longrightarrow PH = \sum_2^P$.