CS70, Fall 2003 **Discussion #6** **Amir Kamil**
UC Berkeley **10/10/03**

Topics: Euler's Theorem, Chinese Remainder Theorem

Consider the following two problems:

1. Compute $2^{110001^{11^{1100001}}} \pmod{23}$.
2. Solve the system of equations:

$$\begin{aligned} 3x &\equiv 5 \pmod 7 \\ 2x &\equiv 6 \pmod 5 \\ x &\equiv 7 \pmod 8. \end{aligned}$$

How would we go about answering these questions? For the first problem, even fast exponentiation would take exponential time. For the second, we could use guess and check, but that quickly becomes intractable for larger problems. In this section, we will see how to solve both problems systematically and efficiently.

1 Euler's Theorem

In class, we saw Fermat's little theorem, which states that $a^{p-1} \equiv 1 \pmod p$ when p is prime and a and p are relatively prime. In addition, we saw a generalization to products of 2 primes, $a^{(p-1)(q-1)} \equiv 1 \pmod{pq}$. Is there a generalization for any n?

The generalization does exist, and relies on *Euler's totient function*, $\phi(n)$. This function is the number of positive integers less than n that are relatively prime to n (with 1 relatively prime to n by definition. For example, $\phi(10) = 4$, since 1, 3, 7, and 9 are relatively prime to 10. For a prime number p, $\phi(p) = p - 1$ since all positive integers less than p are relatively prime to it.

Let's attempt to compute $\phi(n)$ for general $n = pq$ where p and q are distinct primes. Notice that the values $p, 2p, \cdots, (q-1)p$, $q-1$ values total, are not relatively prime to n. In addition, the values $q, 2q, \cdots, (p-1)q$, $p-1$ values total, are also not relatively prime to n. These cover all the positive integers not relatively prime to n, so the number relatively prime to n is $n - 1 - (p-1) - (q-1) = pq - p - q + 1 = (p-1)(q-1)$. Thus $\phi(pq) = (p-1)(q-1)$. The generalization for a product of k distinct primes $n = p_1 p_2 \cdots p_k$ is $\phi(n) = (p_1 - 1)(p_2 - 1) \cdots (p_k - 1)$.

What is $\phi(n)$ for n a product of non-distinct primes? Let's try it for an n that has only two prime factors p and q. Then the values less than n that are not relatively prime to n are $p, 2p, \cdots, (\frac{n}{p} - 1)p$, and the values $q, 2q, \cdots, (\frac{n}{q} - 1)q$, $n/p + n/q - 2$ values total. But we've overcounted, since we counted multiples of pq $(pq, \cdots, (\frac{n}{pq} - 1)pq)$ twice. Thus $\phi(n) = n - 1 - \frac{n}{p} - \frac{n}{q} + 2 + \frac{n}{pq} - 1 = \frac{1}{pq}(npq - nq - np + n) = \frac{n}{pq}(p-1)(q-1)$. For a general n with prime factors $p_1, p_2, \cdots, p_k$, $\phi(n) = (p_1 - 1)(p_2 - 1) \cdots (p_k - 1)\frac{n}{p_1 p_2 \cdots p_k}$.

Then *Euler's theorem* states that if $\gcd(a, n) = 1$, $a^{\phi(n)} \equiv 1 \pmod n$. We can see that this reduces to Fermat's theorem when n is prime, and $a^{(p-1)(q-1)} \equiv 1 \pmod n$ when $n = pq$ is a product of two primes.

We can prove Euler's theorem using Fermat's theorem and the Chinese remainder theorem. Let's do the case where the modulus n is a product of k distinct primes $n = p_1 p_2 \cdots p_k$. Then $\phi(n) = (p_1 - 1)(p_2 - 1) \cdots (p_k - 1)$. Now consider $a^{\phi(n)}$, where a and n are relatively prime. From Fermat's theorem, we know that $a^{\phi(n)} = a^{(p_1-1)\cdots(p_i-1)\cdots(p_k-1)} = (a^{p_i-1})^{(p_1-1)\cdots(p_{i-1}-1)(p_{i+1}-1)\cdots(p_k-1)} \equiv 1 \pmod{p_i}$ for each i. Now we have a set of k equations, so we can apply the Chinese remainder theorem. Trying the solution $a^{\phi(n)} \equiv 1 \pmod n$, we see that it works, and by the Chinese remainder theorem, it is the unique solution.

Using Euler's theorem, we can reduce a^k modulo n, if a and n are relatively prime. Let $k' = k \bmod \phi(n)$, then $k = m\phi(n) + k'$ for some integer m. Then $a^k = a^{m\phi(n)+k'} = (a^{\phi(n)})^m a^{k'} \equiv a^{k'} \pmod n$, using Euler's theorem in the last step. Thus $a^k \equiv a^{k \bmod \phi(n)} \pmod{n}$. We can use this fact to do *blindingly fast exponentation*[1].

Going back to problem 1, we see that $2^{110001^{11^{1100001}}} \equiv 2^{(110001^{11^{1100001}} \bmod 22)} \pmod{23}$. Applying Euler's theorem again, $110001^{11^{1100001}} \equiv 110001^{(11^{1100001} \bmod 10)} \pmod{22}$. Again, $11^{1100001} \equiv 11^{(1100001 \bmod 4)} \equiv 11^1 \equiv 1 \pmod{10}$. Then $110001^{11^{1100001}} \equiv 110001^{(11^{1100001} \bmod 10)} \equiv 110001^1 \equiv 1 \pmod{22}$. Finally, $2^{10001^{11^{1100001}}} \equiv 2^{(110001^{11^{1100001}} \bmod 22)} \equiv 2^1 \equiv 2 \pmod{23}$. Can you see how this algorithm is blindingly fast?

Of course, in order to do blindingly fast exponentiation, we must be able to compute the totient function efficiently. So the question of the day is, is it possible to compute $\phi(n)$ in polynomial (in $\lg n$) time?

Let's assume for the moment that we can compute the totient function, and see where that takes us. Consider $n = pq$, where p and q are prime. Let $\phi(n) = k$. Then, since $\phi(n) = (p-1)(q-1)$, we have the system of equations

$$(p-1)(q-1) = k$$

$$pq = n,$$

where p and q are unknown, and k and n are known. Solving the second for q and substituting into the first, we get $(p-1)(n/p-1) = k$. Multiplying the left side out and then the equation by p, we get $pn - n - p - p^2 = kp$. Rearranging, we get $p^2 + (k-n+1)p + n = 0$. Using the quadratic formula, we solve for p to get $p = \frac{-(k-n+1) \pm \sqrt{(k-n+1)^2 - 4n}}{2}$. Thus, we can solve for p and q, using the value of $\phi(n)$.

But this means we've factored n! And since RSA encryption depends on a product of two primes not being efficiently factorable, we've broken the encryption protocol. Since this contradicts the RSA assumption, it is safe to assume that computing the totient function is intractable.

2 Chinese Remainder Theorem

The Chinese remainder theorem states that a set of equations

$$x \equiv a \pmod{p}$$

$$x \equiv b \pmod{q},$$

where p and q are relatively prime, has exactly one solution modulo pq. But it gives no clue on how to solve the system of equations. Here, we see how to solve these equations systematically.

Since you will write an algorithm to solve such a set of equations in the homework, we will only do a concrete example, problem 2 above. First we reduce the equations to get

$$x \equiv 4 \pmod{7}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 7 \pmod{8}.$$

Now from the first, we know that $x = 7s + 4$ for some integer s. Substituting into the second equation, we get $7s + 4 \equiv 3 \pmod{5}$. Solving for s, we get $s \equiv 2 \pmod{5}$, so $s = 5t + 2$ for some t. Substituting into $x = 7s + 4$, we get $x = 35t + 18$. Now using this in the third equation, we get $35t + 18 \equiv 7 \pmod{8}$. Solving for t, we get $t \equiv 7 \pmod{8}$, so $t = 8u + 7$ for some u. Substituting into $x = 35t + 18$, we get $x = 280u + 263$. Thus $x \equiv 263 \pmod{280}$.

[1] Also called *screaming fast exponentiation.*