Modular Exponentiation and Solving Modular Equations

1 Euler's Totient Function

The **Euler's Totient Function** or **Euler's Phi Function**, $\phi(n)$, counts how many integers in the range $[1, n-1]$ are relatively prime to n. Two numbers are relatively prime if their gcd is equal to 1. This function is especially important when we are performing modular exponentiation due to the following theorem:

Euler's Theorem. *If a and n are relatively prime to each other than:*

$$a^{\phi(n)} \equiv 1 \textit{ (mod n)}$$

The implication of this theorem is that for any integer k, let $r = k - k * \lfloor \frac{k}{\phi(n)} \rfloor$, then $a^k \equiv a^r$ (mod n). This provides a quick way to evaluate modular exponents. Of course, this rely on the ability to determine $\phi(n)$ efficiently. It turns out that there is a closed form formula to do so!

Theorem 1. *Given an integer n, let $n = p_1^{\alpha_1} p_2^{\alpha_2} ... p_k^{\alpha_k}$ be its prime factorization. Then:*

$$\phi(n) = (p_1 - 1)p_1^{\alpha_1 - 1}(p_2 - 1)p_2^{\alpha_2 - 1}...(p_k - 1)p_k^{\alpha_k - 1} = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2})...(1 - \frac{1}{p_k})$$

This theorem can be proven inductively based on the observations that $\phi(p) = p - 1$ for any prime p and $\phi(ab) = \phi(a)\phi(b)$ if a and b are relatively prime. Note that Euler's theorem is the generalized version of the more well known **Fermat's Little Theorem**, which states that if p is a prime and a an integer that is relatively prime to p, then $a^{p-1} \equiv 1$ (mod p).

2 Rabin-Miller Primality Testing

Here, we will discuss a probabilistic algorithm to test primality. First, the algorithm is probabilistic because it may not always return the right answer. More specifically, if the algorithm report n is a composite number, than it is guaranteed that n is not prime. However, when the algorithm report n is prime, it is now always the case that n will be prime. It may be tempting to discard an erroneous algorithm like this, but research has shown that the probability of falsely reporting n is a prime when it's not is very long. Thus, in practice, we simply run the test many times. If at any point, the algorithm reported n is composite, then we are done. Otherwise, we can be fairly sure n is a prime.

Let us investigate how the algorithm work. First, it relies on the following fact:

Theorem 2. *Let p be a prime. Then the equation $x^2 \equiv 1$ (mod p) has only two solutions: $x \equiv 1$ (mod p) and $x \equiv -1$ (mod p)*

Now, we will suppose the p is an odd prime. Then we can write $p - 1 = 2^s * d$ for some integer s, d. Let us randomly choose an integer $a < p$. By Fermat's Little Theorem, we know $a^{p-1} \equiv 1$ (mod p). By above theorem, this means that $a^{2^{s-1}d} \equiv \pm 1$ (mod p). If $a^{2^{s-1}d} \equiv 1$ (mod p), then we can apply the previous theorem again to yield that $a^{2^{s-2}d} \equiv \pm 1$ (mod p). Otherwise, $a^{2^{s-1}d} \equiv -1$ (mod p). Following a similar argument, it then follows that either there exists an integer $0 \leq r < s$ such that $a^{2^r d} \equiv -1$ (mod p) or $a^{2^0 d} \equiv 1$ (mod p).

The Rabin-Miller Test make uses of the contrapositive of the above observation. Suppose we are given an odd integer n and we want to test its primality. If it is composite, then there exists an integer a in which $a^{2^r d} \not\equiv -1 \pmod{p}, \forall 0 \leq r < s$ and $a^{2^0 d} \not\equiv 1 \pmod{p}$. In this case, we call a the **witness** for the compositeness of n.

However, there are no known "good" method of finding witness, so what the algorithm does instead is randomly use an integer from the range between 1 and $n-1$. Furthermore, just because we have found an a such that there exists an integer $0 \leq r < s$ such that $a^{2^r d} \equiv -1 \pmod{p}$ or $a^{2^0 d} \equiv 1 \pmod{p}$ doesn't prove that n is a prime. An example of this (taken from wikipedia) is $n = 221, a = 174$.

```
bool RabinMiller(int p) {
  randomly choose an a < p;
  factor (p-1) = 2^s*d;
  x = a^d % p;
  if (x == 1) return true;
  for (int i = 0; i < s; ++i) {
    if (x == p-1) return true;
    x = (x*x % p);
  }
  return false;
}
```

3 Solving $ax \equiv b$ (mod n)

Let us consider how to solve the eqution $ax \equiv b \pmod{n}$ given a, b, and n. First, let us assume $(a, n) = 1$. Recall from last class that we can used the extended gcd algorithm to find two integers s, t such that $as + nt = 1$. This implies that $as \equiv 1 \pmod{n}$ (we call s the **inverse** of a mod n). Now, if we multiply both side of the equation by s, we get $axs \equiv (as)x \equiv x \equiv bs \pmod{n}$. Thus, the solution we are looking for is $x \equiv bs \pmod{n}$!

Now what if a and n are not relatively prime? Let $g = (a, n)$. Suppose $g \nmid b$, then there is no solution in this case. To see this, note that if such a solution does exists, then by definition, we have $n|(ax - b)$, which implies $(ax - b) = nk$ for some integer k. Now, rearranging the equation, we get $ax - nk = b$. However, the left hand side of the equation is divisible by g while the right hand side is not! Thus, we have found a contradiction.

So the only case left is when $g \,|\, b$. Since b divisible by g, let us write $b = gk$. Again, let us use the extended gcd algorithm to find two integers s, t such that $as + nt = g$. Multiplying both side by s, we get $axs \equiv (as)x \equiv gx \equiv bs \equiv g(ks) \pmod{n}$. So we reach the equation $gx \equiv g(ks) \pmod{n}$. While it may look tempting to cancel out the g on both side of the equation to get $x \equiv ks \pmod{n}$, we cannot do so. Instead, based on a fundamental number theory result, the above equation is equivalent to $x \equiv ks \pmod{\frac{n}{g}}$. It is important to note that we are now taking the mod of a **new number**!

4 Chinese Remainder Theorem

In the previous section, we showed how to reduce an arbitrary equation of the form $ax \equiv b \pmod{n}$ to the form $x \equiv y \pmod{n'}$. Let us now investigate how to solve a system of such equations. The problem is as follows: we are given $a_1, a_2, ..., a_m$ and $n_1, n_2, ..., n_m$ and we want to solve the system:

$$\begin{aligned} x &\equiv a_1 \text{ (mod } n_1) \\ x &\equiv a_2 \text{ (mod } n_2) \\ &\vdots \\ x &\equiv a_m \text{ (mod } n_m) \end{aligned}$$

The **Chinese Remainder Theorem** provides an algorithm to solve such as system. However, note that it is not necessary to solve all m equations simultaneously into a single modular equation. It suffices to combine two modular equations into an equivalent single modular equation. If we can do that, then we can solve a system of m equation by first combining the first two equations, then combine the result and the third equations, and so on. Thus, we will now reduce the problem down into combining $x \equiv a_1$ (mod n_1) and $x \equiv a_2$ (mod n_2). The Chinese Remainder Theorem then says:

Chinese Remainder Theorem. *Given a_1, n_1, a_2, n_2. Let $g = (n_1, n_2)$ and s, t be integers such that $n_1 s + n_2 t = g$. Then a solution to the system of modular equations*

$$\begin{aligned} x &\equiv a_1 \text{ (mod } n_1) \\ x &\equiv a_2 \text{ (mod } n_2) \end{aligned}$$

exists iff $a_1 \equiv a_2$ (mod g). In this case, all solutions satisfy

$$x \equiv n_1 s \lfloor \frac{a_2}{g} \rfloor + n_2 t \lfloor \frac{a_1}{g} \rfloor + r \text{ (mod } \tfrac{n_1 n_2}{g})$$

where $r = a_1 - g\lfloor \frac{a_1}{g} \rfloor = a_2 - g\lfloor \frac{a_2}{g} \rfloor$ (since $a_1 \equiv a_2$ (mod g)).

Before delving deeper into the puzzling expression above, I should remark first that this will look different from the usual Chinese Remainder Theorem. This is because the Chinese Remainder Theorem only handles the case that n_1 and n_2 are relatively prime to each other. This means that $g = 1$ in the above expression, and the theorem simplifies down to the more familiar expression $x \equiv n_1 s a_2 + n_2 t a_1$ (mod $n_1 n_2$).

Here, however, I have given a more general expression that will handle the case $(n_1, n_2) > 1$. Why does the formula work? We can rewrite x as

$$x = n_1 s \lfloor \frac{a_2}{g} \rfloor + n_2 t \lfloor \frac{a_1}{g} \rfloor + r + k \frac{n_1 n_2}{g}$$

for some integer k. Let us now evaluate what x (mod n_1):

$$\begin{aligned} x &\equiv 0 + (n_2 t) \lfloor \frac{a_1}{g} \rfloor + r + k \frac{n_1 n_2}{g} && \text{(mod } n_1) \\ &\equiv g * \lfloor \frac{a_1}{g} \rfloor + (a1 - g \frac{a_1}{g}) + 0 && \text{(mod } n_1) \\ &\equiv a_1 && \text{(mod } n_1) \end{aligned}$$

Similarly, we can double check that $x \equiv a_2$ (mod n_2).