Efficient Modular Exponentiation

R. C. Daileda

February 27, 2018

1 Repeated Squaring

Consider the problem of finding the remainder when a^m is divided by n, where m and n are both is "large." If we assume that $(a, n) = 1$, Euler's theorem allows us to reduce m modulo $\varphi(n)$. But this still leaves us with some (potential) problems:

1. $\varphi(n)$ will still be large as well, so the reduced exponent as well as the order of a modulo n might be too big to handle by previous techniques (e.g. with a hand calculator).
2. Without knowledge of the prime factorization of n, $\varphi(n)$ can be very hard to compute for large n.
3. If $(a, n) > 1$, Euler's theorem doesn't apply so we can't necessarily reduce the exponent in any obvious way.

Here we describe a procedure for finding the remainder when a^m is divided by n that is extremely efficient in all situations. First we provide a bit of motivation.

Begin by expressing m in binary[1]:

$$m = \sum_{j=0}^{N} b_j 2^j, \quad b_j \in \{0,\, 1\} \text{ and } b_N = 1.$$

Notice that

$$a^m = a^{\sum_{j=0}^{N} b_j 2^j} = \prod_{j=0}^{N} a^{b_j 2^j} = \prod_{j=0}^{N} (a^{2^j})^{b_j}$$

and that

$$a^{2^j} = \left(a^{2^{j-1}}\right)^2 = \left(\left(a^{2^{j-2}}\right)^2\right)^2 = \cdots = \underbrace{\left(\left(\left(a^2\right)^2\right)^2 \cdots\right)^2}_{j \text{ squares}}.$$

So we can compute a^m by repeatedly squaring a and then multiplying together the powers for which $b_j = 1$. If we perform these operations pairwise, reducing modulo n at each stage, we will never need to perform modular arithmetic with a number larger than n^2, *no matter how large m is!* We make this procedure explicit in the algorithm described below.

[1]We will discuss how this is done shortly.

Modular Exponentiation by Repeated Squaring. Given $m, n \in \mathbb{N}$ and $a \in \mathbb{Z}$, the following algorithm returns the remainder when a^m is divided by n.

Step 1. Express m in binary:

$$m = \sum_{j=0}^{N} b_j 2^j,$$

where $b_j \in \{0, 1\}$ for all j and $b_N = 1$.

Step 2. Let $a = q_0 n + s_0$ with $0 \le s_0 < n$ and, for $1 \le i \le N$, define s_i through the equation $s_{i-1}^2 = q_i n + s_i$ with $0 \le s_i < n$. [Repeatedly square a and reduce modulo n at each stage.]

Step 3. Let $P_0 = s_0^{b_0}$ and, for $1 \le i \le N$, define P_i through the equation $P_{i-1} s_i^{b_i} = \ell_i n + P_i$ with $0 \le P_i < n$. [Construct the partial power one factor at a time and reduce modulo n at each stage.]

Step 4. Return P_N.

Proof. According to Step 3 we have

$$\begin{aligned} P_N &\equiv P_{N-1} s_N^{b_N} \pmod{n} \\ &\equiv P_{N-2} s_{N-1}^{b_{N-1}} s_N^{b_N} \pmod{n} \\ &\equiv P_{N-3} s_{N-2}^{b_{N-2}} s_{N-1}^{b_{N-1}} s_N^{b_N} \pmod{n} \\ &\vdots \\ &\equiv s_0^{b_0} s_1^{b_1} s_2^{b_2} \cdots s_N^{b_N} \pmod{n} \end{aligned}$$

and by Step 2

$$\begin{aligned} s_1 &\equiv s_0^2 \pmod{n}, \\ s_2 &\equiv s_1^2 \equiv s_0^{2^2} \pmod{n}, \\ s_3 &\equiv s_2^2 \equiv s_0^{2^3} \pmod{n}, \\ &\vdots \\ s_i &\equiv s_0^{2^i} \pmod{n}. \end{aligned}$$

Hence, since $a \equiv s_0 \pmod{n}$,

$$P_N \equiv s_0^{b_0} s_1^{b_1} s_2^{b_2} \cdots s_N^{b_N} \equiv s_0^{b_0} s_0^{b_1 \cdot 2} s_0^{b_2 \cdot 2^2} \cdots s_0^{b_N \cdot 2^N} \equiv s_0^{\sum_{j=0}^{N} b_j 2^j} \equiv a^m \pmod{n}.$$

□

As noted above, once we have reduced a modulo n, at no point in the algorithm will we ever be dealing with a number larger than n^2, regardless of the size of m. This is central to the algorithm's efficiency. The only other question to be addressed is how efficient base conversions are. Before we touch on that, an example is in order.

Example 1. Find the remainder when 321^{12345} is divided by 54321.

Since it is not difficult to see that $54321 = 3 \cdot 19 \cdot 953$, it might not be a bad idea to reduce the exponent modulo $\varphi(54321) = 2 \cdot 18 \cdot 952 = 34272$. But: (a) 12345 is already reduced and (b) $(321, 54321) = 3$ so Euler's theorem doesn't apply anyway. So we simply apply the Repeated Squaring algorithm without any preliminary simplification.

Maple returns the binary expansion of 12345 almost instantly:

$$1 + 2^3 + 2^4 + 2^5 + 2^{12} + 2^{13}.$$

We now repeatedly square 321, reducing modulo 54321 at each stage:

$$\begin{aligned}
321^2 &\equiv 48720 \pmod{54321},\\
321^{2^2} &\equiv 48720^2 \equiv 27984 \pmod{54321},\\
321^{2^3} &\equiv 27984^2 \equiv 12720 \pmod{54321},\\
321^{2^4} &\equiv 12720^2 \equiv 30462 \pmod{54321},\\
321^{2^5} &\equiv 30462^2 \equiv 22122 \pmod{54321},\\
321^{2^6} &\equiv 22122^2 \equiv 4995 \pmod{54321},\\
321^{2^7} &\equiv 4995^2 \equiv 16686 \pmod{54321},\\
321^{2^8} &\equiv 16686^2 \equiv 27471 \pmod{54321},\\
321^{2^9} &\equiv 27471^2 \equiv 28509 \pmod{54321},\\
321^{2^{10}} &\equiv 28509^2 \equiv 12279 \pmod{54321},\\
321^{2^{11}} &\equiv 12279^2 \equiv 33066 \pmod{54321},\\
321^{2^{12}} &\equiv 33066^2 \equiv 41589 \pmod{54321},\\
321^{2^{13}} &\equiv 41589^2 \equiv 9960 \pmod{54321}.
\end{aligned}$$

Finally we multiply together, a pair at a time, only those powers of 2 that occur in the binary expansion of the exponent 12345:

$$\begin{aligned}
&321 \cdot 321^{2^3} \equiv 321 \cdot 12720 \equiv 9045 \pmod{54321},\\
&321 \cdot 321^{2^3} \cdot 321^{2^4} \equiv 9045 \cdot 30462 \equiv 12678 \pmod{54321},\\
&321 \cdot 321^{2^3} \cdot 321^{2^4} \cdot 321^{2^5} \equiv 12678 \cdot 22122 \equiv 3393 \pmod{54321},\\
&321 \cdot 321^{2^3} \cdot 321^{2^4} \cdot 321^{2^5} \cdot 321^{2^{12}} \equiv 3393 \cdot 41589 \equiv 39840 \pmod{54321},\\
&321^{12345} = 321 \cdot 321^{2^3} \cdot 321^{2^4} \cdot 321^{2^5} \cdot 321^{2^{12}} \cdot 321^{2^{13}} \equiv 39840 \cdot 9960 \equiv 45816 \pmod{54321}.
\end{aligned}$$

So that the remainder is $\boxed{45816}$. ♦

We can actually improve the efficiency of the Repeated Squaring algorithm by combining Steps 2 and 3, computing the partial products P_i and repeated squares s_i simultaneously, then discarding each s_i after we've used it, rather than store them all.

Improved Modular Exponentiation by Repeated Squaring. Given $m, n \in \mathbb{N}$ and $a \in \mathbb{Z}$, the following algorithm returns the remainder when a^m is divided by n.

Step 1. Express m in binary:

$$m = \sum_{j=0}^{N} b_j 2^j,$$

where $b_j \in \{0, 1\}$ for all j and $b_N = 1$.

Step 2. Let $a = q_0 n + s_0$ and $P_0 = s_0^{b_0}$.

Step 3. For $1 \leq i \leq N$, given s_{i-1} and P_{i-1}, define s_i through the equation $s_{i-1}^2 = q_i n + s_i$ with $0 \leq s_i < n$ and P_i through the equation $P_{i-1} s_i^{b_i} = \ell_i n + P_i$ with $0 \leq P_i < n$. Discard s_{i-1} and P_{i-1}.

Step 4. Return P_N.

Proof. The proof is the same. □

It is worth noting that, while tedious, all of the computations above could easily have been implemented using only a four function calculator. The real advantage of the algorithm can be seen, however, when it is implemented on a computer. For example, suppose we boost the exponent of the preceding example to 123456789. It takes Maple (running on a 3.3 GHz Intel i5 Mac) about 6 seconds to report that the remainder when $321^{123456789}$ is divided by 54321 is 37488. Whereas an implementation of the Repeated Squaring algorithm in Maple (on the same computer) yields the same result almost instantaneously. Both SAGE and PARI behave the same way automatically, i.e. are hard-coded to use (some variant of) the Repeated Squaring algorithm for modular exponentiation.

2 Primality Tests Involving Exponentiation

Recall Fermat's little theorem: if p is prime and $(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$. Consider the contrapositive: if there exists an $a \in \mathbb{Z}$ such that $(a, p) = 1$ and $a^{p-1} \not\equiv 1 \pmod{p}$, then p is composite. Because, as we have just seen, modular exponentiation is quite efficient, this will provide our first truly useful primality test.

Fermat Primality Test. Given an odd number n, the following algorithm either proves that n is composite or returns a congruence of the form $a^{n-1} \equiv 1 \pmod{n}$, $(a, n) = 1$.

Step 1. Randomly choose an integer $2 \leq a < n$.

Step 2. Use the EA to compute $d = (a, n)$. If $d > 1$, we have a nontrivial factor of n, so we terminate the algorithm and output "composite" (and d for good measure).[a]

Step 3. Compute the remainder r when a^{n-1} is divided by n (e.g. by using Repeated Squaring).

Step 4. If $r \neq 1$, output "composite." If $r = 1$, output the congruence $a^{n-1} \equiv 1 \pmod{n}$.

[a]We don't really expect this to happen often.

Remark 1.

- The Fermat test (as I've called it) is really a compositeness test. And since Fermat's little theorem isn't a biconditional result, when the test fails we can't conclude anything concrete. It might be that n is prime, or n might actually be composite. Composite numbers that satisfy $a^{n-1} \equiv 1 \pmod{n}$ for some $(a, n) = 1$ are called *pseudoprimes to the base* a.
- If an integer n fails the Fermat test frequently enough, we might start to believe it is actually prime. However, there exist n that are pseudoprimes to every base $(a, n) = 1$. Such n are called *Carmichael numbers*; the smallest is 561. Carmichael numbers are what prevent the Fermat Primality Test from actually being a primality test! A long standing open question was just how many Carmichael numbers exist. It was settled in 1994 by Erdös, Granville and Pomerance: there are infinitely many. D'oh!
- Except in the (exceedingly rare case) that $d > 1$ in Step 2, even in the Fermat test gives an affirmative result, it doesn't provide any information on what the factors of n actually are.

▼

Example 2. Prove that

$$\begin{aligned} n =& 21954697995095410853358122600332847218651088538566 \\ & 91125738388026532740045150707008069155261775991153 \end{aligned}$$

(100 digits) is composite.

Using the (Improved) Repeated Squaring algorithm we find (almost instantly) that

$$\begin{aligned} 2^{n-1} \equiv& 81841814842480266898388843962502616548343355500389 \\ & 39031384387917696321692521689702864546776183558039 \pmod{n}. \end{aligned}$$

Since we did not get 1, n must be composite. As noted above, we do not have any information as to how n actually factors. ♦

Appendix: Base B Representations of Natural Numbers

As an aside, we now address the issue of the expression of natural numbers in different bases. Let $B \geq 2$ be the given *base* and let $D_B = \{0, 1, 2, \ldots, B-1\}$ denote the set of base B "digits." The following result is often tacitly assumed, but rarely proven.

Theorem 1. *Let $n \in \mathbb{N}$. Then n has a unique* base B expansion

$$n = \sum_{j=0}^{N} d_j B^j, \quad N \in \mathbb{N}_0, \quad d_j \in D_B, \quad d_N \neq 0. \tag{1}$$

Proof. **Existence.** We induct on n. Clearly $1 \cdot B^0$ is a base B expansion of $n = 1$, since $B \geq 2$. Now assume that for some $n \geq 1$, all natural numbers $k \leq n$ have base B expansions. Let

$N \in \mathbb{N}_0$ satisfy $B^N \le n+1 < B^{N+1}$ and use the division algorithm to write $n+1 = qB^N + r$ with $0 \le r < B^N$. If $q \ge B$, then $B^{N+1} \le qB^N \le qB^N + r = n+1$, a contradiction. Therefore $q \in D_B$. And if $q = 0$, then $n+1 = r < B^N$, another contradiction. So if we set $d_N = q$ then $d_N \in D_B$ and $d_N \neq 0$.

If $r = 0$ then $n+1 = q_N B^N$ is a base B expansion. Otherwise, $0 < r < B^N$ implies $0 < r \le n$ so that r has a base B expansion by the inductive hypothesis. If we write

$$r = \sum_{j=0}^{M} d_j B^j$$

with $M \in \mathbb{N}_0$, $d_j \in D_B$, and $d_M \neq 0$, then since $n+1 = d_N q^N + r$ it suffices to show that $M < N$. But since $d_M \ge 1$, we see that $B^M \le r < B^N$, which implies $M < N$. Thus

$$n+1 = \sum_{j=0}^{M} d_j B^j + d_N B^N$$

is a base B expansion for $n+1$ and the general existence is established by induction.

Uniqueness. We make the following claim: if $a_j \in \mathbb{Z}$ satisfy $|a_j| < B$ and $m \in \mathbb{N}_0$, then

$$\sum_{j=0}^{m} a_j B^j = 0 \quad \Rightarrow \quad a_j = 0 \text{ for all } j.$$

We prove the claim by induction on m, the case $m = 0$ being trivial. So suppose it holds for some $m \ge 0$ and assume

$$\sum_{j=0}^{m+1} a_j B^j = 0$$

with $|a_j| < B$. We then have

$$|a_{m+1} B^{m+1}| = \left| \sum_{j=0}^{m} a_j B^j \right| \le (B-1) \sum_{j=0}^{m} B^j = B^{m+1} - 1 < B^{m+1}$$

which implies $|a_{m+1}| < 1$. As a_{m+1} is an integer this can only occur if $a_{m+1} = 0$. But the our assumption becomes

$$\sum_{j=0}^{m} a_j B^j = 0$$

which implies $a_j = 0$ for all other j as well by the inductive hypothesis. So the result holds for $m+1$ if it holds for m and we've established the claim.

Uniqueness of base B expansions is now easy to establish. If

$$n = \sum_{j=0}^{N} d_j B^j, \;\; N \in \mathbb{N}_0, \;\; d_j \in D_B, \;\; d_N \neq 0$$

first note that since $d_N \neq 0$

$$B^N \le n \le (B-1) \sum_{j=0}^{N} B^j = B^{N+1} - 1 < B^{N+1}$$

that is N is uniquely determined by the condition

$$B^N \leq n < B^{N+1}.$$

So any other base B expansion of N must have the form

$$n = \sum_{j=0}^{N} e_j B^j, \ \ e_j \in D_B, \ \ e_N \neq 0.$$

If we subtract these expansions and apply the claim we immediately conclude that $d_j = e_j$ for all j, as required. □

Remark 2.

- The base B expansion (1) of n is rather cumbersome so we introduce the *base B place-value notation* $n = (d_N d_{N-1} \cdots d_2 d_1 d_0)_B$. When $B = 10$ we omit the parentheses and the subscript altogether.
- Although we are used to thinking of integers in terms of their base 10 (decimal) expansions, these are simply *representations* of the integers, not the integers themselves. Integers are abstract entities and base B expansions simply give us a convenient way to work with them.
- The existence portion of the proof actually gives and algorithm for finding the base B expansion of n. First, find N so that $B^N \leq n < B^{N+1}$. Perform the division algorithm: $n = qB^N + r$. Then $q = d_N$. Now replace q with r and repeat. Continue until $r \in D_B$, at which point you've found d_0.

▼

Example 3. Find the binary (base 2) expansion of 3849.

Since $2^{11} \leq 3849 < 2^{12}$, this is our starting point. And since the only digits are 0 and 1, we simply subtract to find the remainder: $3849 - 2^{11} = 1801$. Now $2^{10} \leq 1801 < 2^1 1$ and $1801 - 2^{10} = 777$. Again $2^9 \leq 777 < 2^{10}$ and $777 - 2^9 = 265$. This time $2^8 \leq 265 < 2^9$ and $265 - 2^8 = 9$. Finally, $9 = 2^3 + 1$ so we find that

$$\boxed{3849 = 2^{11} + 2^{10} + 2^9 + 2^8 + 2^3 + 1 = (111100001001)_2}$$

♦

Example 4. Find the ternary (base 3) expansion of 3849.

Now we have $3^7 \leq 3849 < 3849$ and the division algorithm yields $3849 = 1 \cdot 3^7 + 1662$. Next $3^6 \leq 1662 < 3^7$ and $1662 = 2 \cdot 3^6 + 204$. Now $3^4 \leq 204 < 3^5$ and $204 = 2 \cdot 3^4 + 42$. At this point we just proceed manually: $42 = 27 + 15 = 27 + 9 + 6 = 27 + 9 + 2 \cdot 3$. Thus

$$\boxed{3849 = 3^7 + 2 \cdot 3^6 + 2 \cdot 3^4 + 3^3 + 3^2 + 2 \cdot 3 = (12021120)_3}$$

♦

Remark 3. *Kummer's theorem* states that for m, $n \in \mathbb{N}_0$ satisfying $m \leq n$, the exact power of p dividing $\binom{n}{m}$ is the number of carries when m is added to $n - m$ in base p. Note that this provides an alternate proof of the divisibility of the binomial coefficients $\binom{p}{k}$ by p when $1 \leq k \leq p - 1$. ▼