FERMAT'S LITTLE THEOREM

KEITH CONRAD

1. Introduction

When we compute the powers of nonzero numbers modulo a prime, something striking happens when powers of different numbers are compared to each other: they all become 1 when the exponent is $p-1$.

Example 1.1. The tables below show the powers of nonzero numbers mod 5 and nonzero numbers mod 7. In the first table all fourth powers are 1, and in the second table all sixth powers are 1.

k	1	2	3	4
$1^k \bmod 5$	1	1	1	1
$2^k \bmod 5$	2	4	3	1
$3^k \bmod 5$	3	4	2	1
$4^k \bmod 5$	4	2	4	1

k	1	2	3	4	5	6
$1^k \bmod 7$	1	1	1	1	1	1
$2^k \bmod 7$	2	4	1	2	4	1
$3^k \bmod 7$	3	2	6	4	5	1
$4^k \bmod 7$	4	2	1	4	2	1
$5^k \bmod 7$	5	4	6	2	3	1
$6^k \bmod 7$	6	1	6	1	6	1

These tables illustrate a fundamental property of prime numbers due to Fermat.

Theorem 1.2 (Fermat)**.** *For a prime number p and every $a \not\equiv 0 \bmod p$, $a^{p-1} \equiv 1 \bmod p$.*

This is called *Fermat's little theorem*. After proving it we will indicate how it can be turned into a method of proving numbers are composite without having to find a factorization for them.

2. Proof of Fermat's Little Theorem

The proof of Fermat's little theorem relies on a simple but clever idea: write down the same list in two different ways and then compare their products.

Proof. We have a prime p and an arbitrary $a \not\equiv 0 \bmod p$. To show $a^{p-1} \equiv 1 \bmod p$, consider non-zero integers modulo p in the standard range:

$$S = \{1, 2, 3, \ldots, p-1\}.$$

We will compare S with the set obtained by multiplying the elements of S by a:

$$aS = \{a, a \cdot 2, a \cdot 3, \ldots, a(p-1)\}.$$

The elements of S represent the nonzero numbers modulo p and (the key point!) the elements of aS *also* represent the nonzero numbers modulo p. That is, every nonzero number mod p is congruent to exactly one number in aS. Indeed, for any $b \not\equiv 0 \bmod p$, the congruence $ax \equiv b \bmod p$ has a solution x since $a \bmod p$ is invertible, and necessarily $x \not\equiv 0 \bmod p$ (since $b \not\equiv 0 \bmod p$). Adjusting x modulo p to lie between 1 and $p-1$ we have $x \in S$, so $ax \in aS$ and thus $b \bmod p$ is represented by an element of aS. Different elements of aS don't represent the same number mod p since $ax \equiv ay \bmod p \Longrightarrow x \equiv y \bmod p$, and different elements of S are not congruent mod p.

Since S and aS become the same thing when reduced modulo p, the product of the numbers in each set must be the same modulo p:

$$1 \cdot 2 \cdot 3 \cdot \cdots \cdot (p-1) \equiv a(a \cdot 2)(a \cdot 3) \cdots (a(p-1)) \bmod p.$$

Pulling the $p-1$ copies of a to the front of the product on the right, we get

$$1 \cdot 2 \cdot 3 \cdot \cdots \cdot (p-1) \equiv a^{p-1}(1 \cdot 2 \cdot 3 \cdot \cdots \cdot (p-1)) \bmod p.$$

Now we cancel each of $1, 2, 3, \ldots, p-1$ on both sides (since they are all invertible modulo p) and we are left with $1 \equiv a^{p-1} \bmod p$. □

Let's illustrate the idea behind this proof when $p = 7$.

Example 2.1. When $p = 7$, $S = \{1, 2, 3, 4, 5, 6\}$. Taking $a = 2$, if we double the elements of S we get $2S = \{2, 4, 6, 8, 10, 12\}$. This is not the same set of integers as S, but $2S$ turns into S when we reduce it mod 7:

$$2 \equiv 2,\ 4 \equiv 4,\ 6 \equiv 6,\ 8 \equiv 1,\ 10 \equiv 3,\ 12 \equiv 5.$$

Similarly, $3S = \{3, 6, 9, 12, 15, 18\}$ and, modulo 7,

$$3 \equiv 3,\ 6 \equiv 6,\ 9 \equiv 2,\ 12 \equiv 5,\ 15 \equiv 1,\ 18 \equiv 4.$$

For any $a \not\equiv 0 \bmod 7$, the sets $\{1, 2, 3, 4, 5, 6\}$ and $\{a, 2a, 3a, 4a, 5a, 6a\}$ turn into the same list mod 7, so their products are the same modulo 7:

$$1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \equiv a \cdot 2a \cdot 3a \cdot 4a \cdot 5a \cdot 6a \equiv a^6(1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6) \bmod 7.$$

Canceling the common factors 1, 2, 3, 4, 5, and 6 from both sides, since they are all nonzero mod 7, we are left with $1 \equiv a^6 \bmod 7$.

Remark 2.2. In the proof we were led to $1 \cdot 2 \cdot 3 \cdots (p-1) = (p-1)!$ modulo p, but we did *not* have to calculate it at all, since after creating this product on both sides of a congruence we canceled it term by term. It turns out there is a simple formula for this product: $(p-1)! \equiv -1 \bmod p$ when p is prime. That is called Wilson's theorem. It is irrelevant to the proof of Fermat's little theorem.

3. Using Fermat's Little Theorem to Prove Compositeness

A crucial aspect of Fermat's little theorem is that it is a property of *every* $a \not\equiv 0 \bmod p$. To emphasize that, let's rewrite Fermat's little theorem like this:

If p is a prime number then $a^{p-1} \equiv 1 \bmod p$ for all $a \not\equiv 0 \bmod p$.

The expression a^{p-1} in the congruence still makes sense if we replace p with any positive integer m, so the contrapositive of Fermat's little theorem says:

If m is a positive integer and $a^{m-1} \not\equiv 1 \bmod m$ for *some* $a \not\equiv 0 \bmod m$ then m is not prime.

This suggests the potential of proving a number $m \geq 2$ is composite without having to factor it: just find a single $a \not\equiv 0 \bmod m$ for which $a^{m-1} \not\equiv 1 \bmod m$.

Example 3.1. Let $m = 48703$. Since $2^{m-1} \equiv 11646 \not\equiv 1 \bmod m$, the number 48703 must be composite. We know that without having any idea of how to factor 48703 into a product of smaller numbers. Of course you can use a computer to rapidly determine that the prime factorization of m is $113 \cdot 431$, but that is a separate issue.

Example 3.2. Let $m = 80581$. Since $2^{m-1} \equiv 1 \bmod m$, there is no contradiction. Maybe m is prime. But $3^{m-1} \equiv 76861 \not\equiv 1 \bmod m$, so $a = 3$ would violate Fermat's little theorem if m *were* prime, so it can't be prime. The number 80581 must be composite but the reason we know this does not tell us how to factor it.

These examples illustrate a point that is at first hard to believe: proving a number is composite and factoring a number in a nontrivial way are not the same task. It is often easier to prove a number *has* a nontrivial factorization than it is to *find* a nontrivial factorization. (Cryptographic protocols used every time you send information over the internet depend on this distinction.) In practice composite numbers having hundreds of digits will have their compositeness revealed by the above method after testing just a few values of a on a computer, and there are large numbers known to be composite but for which no nontrivial factor is known.

A reason it is computationally efficient to check $a^{m-1} \not\equiv 1 \bmod m$ is that when you ask a computer to calculate a large power of a number, such as a^{48702} in Example 3.1, the computer is *not* carrying out anything close to 48000 multiplications. There is a much faster way! By writing the exponent 48702 in binary, the calculation of a^{48702} turns into repeated squaring and can be done with far fewer multiplications than the size of the exponent.

Example 3.3. Since

$$48702 = 2 + 2^2 + 2^3 + 2^4 + 2^5 + 2^9 + 2^{10} + 2^{11} + 2^{12} + 2^{13} + 2^{15},$$

we can write

$$a^{48702} = a^2 a^{2^2} a^{2^3} a^{2^4} a^{2^5} a^{2^9} a^{2^{10}} a^{2^{11}} a^{2^{12}} a^{2^{13}} a^{2^{15}}, \tag{3.1}$$

which is a product of 11 numbers. Each a^{2^k} is the result of squaring a successively k times. If we compute a^{2^k} for $k = 1, 2, \ldots, 15$ and save that data, then the number of multiplications we need to get a^{48702} is quite small: 15 squarings to get from a^2 to $a^{2^{15}}$, and then 10 multiplications of the different values of a^{2^k} on the right side of (3.1). That is a total of just $15 + 10 = 25$ multiplications to determine a^{48702}.

Remark 3.4. How many multiplications are needed to compute a^N? If $2^d \leq N < 2^{d+1}$ then writing N in binary makes a^N a product of terms from $\{a, a^2, a^4, \ldots, a^{2^d}\}$. We need d repeated squarings to get each of these terms and at most d multiplications of these terms to get a^N, which is a total of *at most* $2d \leq 2\log_2 N$ multiplications. This is much less than N when N is even moderately big. If $N = 48702$ then $2\log_2 N \approx 31.14$, while we found a^{48702} in Example 3.3 needs 25 multiplications, showing how $2\log_2 N$ works as an upper bound on the number of required multiplications.

Another aspect that keeps computations under control is that we are interested not in a^{m-1} itself, but in $a^{m-1} \bmod m$. When m is large, calculating a^{m-1} as a raw integer and then reducing it mod m takes much longer than computing $(a \bmod m)^{m-1}$: doing repeated

squaring mod m and reducing intermediate products mod m every time keeps the output from ever getting much larger than the size of m itself. For example, using the computer algebra package Sage to determine $11^{56052360} \bmod 56052361$, the calculation of $11^{56052360}$ in $\mathbf{Z}$ followed by reduction modulo 56052361 took 2.210 seconds while the calculation of $(11 \bmod 56052361)^{56052360}$ took 17 microseconds (that's 17 millionths of a second). The first calculation takes 130,000 times as long as the second one.