Fermat primality test - Wikipedia, the free encyclopedia

The **Fermat primality test** is a probabilistic test to determine whether a number is a probable prime.

Concept[edit]

Fermat's little theorem states that if *p* is prime and

$$0 < a < p$$

, then

$$a^{p-1} \equiv 1 \pmod{p}.$$

If we want to test whether *p* is prime, then we can pick random *a*'s in the interval and see whether the equality holds. If the equality does not hold for a value of *a*, then *p* is composite. If the equality does hold for many values of *a*, then we can say that *p* is probably prime.

It might be in our tests that we do not pick any value for *a* such that the equality fails. Any *a* such that

$$a^{n-1} \equiv 1 \pmod{n}$$

when *n* is composite is known as a *Fermat liar*. Vice versa, in this case *n* is called Fermat pseudoprime to base *a*.

If we do pick an *a* such that

$$a^{n-1} \not\equiv 1 \pmod{n}$$

then *a* is known as a *Fermat witness* for the compositeness of *n*.

Example[edit]

Suppose we wish to determine whether *n* = 221 is prime. Randomly pick 1 < *a* < 221, say *a* = 38. We check the above equality and find that it holds:

$$a^{n-1} = 38^{220} \equiv 1 \pmod{221}.$$

Either 221 is prime, or 38 is a Fermat liar, so we take another *a*, say 24:

$$a^{n-1} = 24^{220} \equiv 81 \not\equiv 1 \pmod{221}.$$

So 221 is composite and 38 was indeed a Fermat liar.

Algorithm and running time[edit]

The algorithm can be written as follows:

Inputs: *n*: a value to test for primality, *n*>3; *k*: a parameter that determines the number of times to test for primality

Output: *composite* if *n* is composite, otherwise *probably prime*

Repeat *k* times:

Pick *a* randomly in the range [2, *n* − 2]

If

$$a^{n-1} \not\equiv 1 \pmod n$$

, then return *composite*

If composite is never returned: return *probably prime*

The *a* values 1 and *n*-1 are not used as the equality holds for all *n* and all odd *n* respectively, hence testing them adds no value.

Using fast algorithms for modular exponentiation, the running time of this algorithm is O(*k* × $\log^2 n$ × log log *n* × log log log *n*), where *k* is the number of times we test a random *a*, and *n* is the value we want to test for primality.

Flaw[edit]

There are infinitely many values of

$$n$$

(known as Carmichael numbers) for which all values of

$$a$$

for which

$$gcd(a, n) = 1$$

are Fermat liars. For these numbers, repeated application of the Fermat primality test performs the same as a simple random search for factors. While Carmichael numbers are substantially rarer than prime numbers,[1] there are enough of them that Fermat's primality test is not often used in the above form. Instead, other more powerful extensions of the Fermat test, such as Baillie-PSW, Miller-Rabin, and Solovay-Strassen are more commonly used.

In general, if

$$n$$

is not a Carmichael number then at least half of all

$$a \in (\mathbb{Z}/n\mathbb{Z})^*$$

are Fermat witnesses. For proof of this, let

a
be a Fermat witness and

a_1

,

a_2

, ...,

a_s

be Fermat liars. Then

$$(a \cdot a_i)^{n-1} \equiv a^{n-1} \cdot a_i^{n-1} \equiv a^{n-1} \not\equiv 1 \pmod{n}$$

and so all

$a \times a_i$

for

$i = 1, 2, ..., s$

are Fermat witnesses.

Applications[edit]

As mentioned above, most applications use a Miller-Rabin or Baillie-PSW test for primality. Sometimes a Fermat test (along with some trial division by small primes) is performed first to improve performance. GMP since version 3.0 uses a base-210 Fermat test after trial division and before running Miller-Rabin tests. Libgcrypt uses a similar process with base 2 for the Fermat test, but OpenSSL does not.

In practice with most big number libraries such as GMP, the Fermat test is not noticeably faster than a Miller-Rabin test, and can be slower for many inputs.[2]

As an exception, OpenPFGW uses only the Fermat test for probable prime testing. The program is typically used with multi-thousand digit inputs with a goal of maximum speed with very large inputs. Another well known program that relies only on the Fermat test is PGP where it is only used for testing of self-generated large random values (an open source counterpart, GNU Privacy Guard, uses a Fermat pretest followed by Miller-Rabin tests).