Fermat's factorization method

From Wikipedia, the free encyclopedia

Fermat's factorization method, named after Pierre de Fermat, is based on the representation of an odd integer as the difference of two squares:

$$N = a^2 - b^2.$$

That difference is algebraically factorable as $(a+b)(a-b)$; if neither factor equals one, it is a proper factorization of *N*.

Each odd number has such a representation. Indeed, if $N = cd$ is a factorization of *N*, then

$$N = \left(\frac{c+d}{2}\right)^2 - \left(\frac{c-d}{2}\right)^2$$

Since *N* is odd, then *c* and *d* are also odd, so those halves are integers. (A multiple of four is also a difference of squares: let *c* and *d* be even.)

In its simplest form, Fermat's method might be even slower than trial division (worst case). Nonetheless, the combination of trial division and Fermat's is more effective than either.

Contents [hide]

Basic method [edit]

One tries various values of *a*, hoping that $a^2 - N = b^2$, a square.

```
FermatFactor(N): // N should be odd
    a ← ceil(sqrt(N))
    b2 ← a*a - N
    while b2 isn't a square:
        a ← a + 1 // equivalently: b2 ← b2 + 2*a + 1
        b2 ← a*a - N // a ← a + 1
    endwhile
    return a - sqrt(b2) // or a + sqrt(b2)
```

For example, to factor $N = 5959$, the first try for *a* is the square root of 5959 rounded up to the next integer, which is 78. Then, $b^2 = 78^2 - 5959 = 125$. Since 125 is not a square, a second try is made by increasing the value of *a* by 1. The second attempt also fails, because 282 is again not a square.

Try:	1	2	3
a	78	79	80
***b*²**	125	282	441
b	11.18	16.79	21

The third try produces the perfect square of 441. So, $a = 80$, $b = 21$, and the factors of 5959 are $a - b = 59$ and $a + b = 101$.

Suppose N has more than two prime factors. That procedure first finds the factorization with the least values of *a* and *b*. That is, $a + b$ is the smallest factor ≥ the square-root of *N*, and so $a - b = N/(a + b)$ is the largest factor ≤ root-*N*. If the procedure finds $N = 1 \cdot N$, that shows that *N* is prime.

For $N = cd$, let *c* be the largest subroot factor. $a = (c + d)/2$, so the number of steps is approximately $(c + d)/2 - \sqrt{N} = (\sqrt{d} - \sqrt{c})^2/2 = (\sqrt{N} - c)^2/2c$.

If *N* is prime (so that $d = 1$), one needs $O(N)$ steps. This is a bad way to prove primality. But if *N* has a factor close to its square root, the method works quickly. More precisely, if *c* differs less than $(4N)^{1/4}$ from $\sqrt{N}$, the method requires only one step; this is independent of the size of *N*.[citation needed]

Fermat's and trial division [edit]

Consider trying to factor the prime number *N* = 2345678917, but also compute *b* and *a* − *b* throughout. Going up from $\sqrt{N}$, we can tabulate:

a	48,433	48,434	48,435	48,436
b²	76,572	173,439	270,308	367,179
b	276.7	416.5	519.9	605.9
a − b	48,156.3	48,017.5	47,915.1	47,830.1

In practice, one wouldn't bother with that last row, until *b* is an integer. But observe that if *N* had a subroot factor above $a - b = 47830.1$, Fermat's method would have found it already.

Trial division would normally try up to 48,432; but after only four Fermat steps, we need only divide up to 47830, to find a factor or prove primality.

This all suggests a combined factoring method. Choose some bound $c > \sqrt{N}$; use Fermat's method for factors between $\sqrt{N}$ and c. This gives a bound for trial division which is $c - \sqrt{c^2 - N}$. In the above example, with $c = 48436$ the bound for trial division is 47830. A reasonable choice could be $c = 55000$ giving a bound of 28937.

In this regard, Fermat's method gives diminishing returns. One would surely stop before this point:

a	60,001	60,002
b²	1,254,441,084	1,254,561,087
b	35,418.1	35,419.8
a − b	24,582.9	24,582.2

Sieve improvement [edit]

It is not necessary to compute all the square-roots of $a^2 - N$, nor even examine all the values for a. Consider the table for $N = 2345678917$:

a	48,433	48,434	48,435	48,436
b²	76,572	173,439	270,308	367,179
b	276.7	416.5	519.9	605.9

One can quickly tell that none of these values of b2 are squares. Squares are always congruent to 0, 1, 4, 5, 9, 16 modulo 20. The values repeat with each increase of a by 10. In this example, N is 17 mod 20, so subtracting 17 mod 20 (or adding 3), $a^2 - N$ produces 3, 4, 7, 8, 12, and 19 modulo 20 for these values. It is apparent that only the 4 from this list can be a square. Thus, a^2 must be 1 mod 20, which means that a is 1 or 9 mod 10; it will produce a b^2 which ends in 4 mod 20 and, if square, b will end in 2 or 8 mod 10.

This can be performed with any modulus. Using the same $N = 2345678917$,

modulo 16:	Squares are	0, 1, 4, or 9
	N mod 16 is	5
	so a^2 can only be	9
	and a must be	3 or 5 or 11 or 13 modulo 16
modulo 9:	Squares are	0, 1, 4, or 7
	N mod 9 is	7

so a^2 can only be 7

and a must be 4 or 5 modulo 9

One generally chooses a power of a different prime for each modulus.

Given a sequence of *a*-values (start, end, and step) and a modulus, one can proceed thus:

```
FermatSieve(N, astart, aend, astep, modulus)
    a ← astart
    do modulus times:
        b2 ← a*a - N
        if b2 is a square, modulo modulus:
            FermatSieve(N, a, aend, astep * modulus, NextModulus)
        endif
        a ← a + astep
    enddo
```

But the recursion is stopped when few *a*-values remain; that is, when (aend-astart)/astep is small. Also, because *a*'s step-size is constant, one can compute successive b2's with additions.

Multiplier improvement [edit]

Fermat's method works best when there is a factor near the square-root of *N*.

If the approximate ratio of two factors (d/c) is known, then the rational number v/u can picked near that value. $Nuv = cv \cdot du$, and the factors are roughly equal: Fermat's, applied to *Nuv*, will find them quickly. Then $\gcd(N, cv) = c$ and $\gcd(N, du) = d$. (Unless *c* divides *u* or *d* divides *v*.)

Generally, if the ratio is not known, various u/v values can be tried, and try to factor each resulting *Nuv*. R. Lehman devised a systematic way to do this, so that Fermat's plus trial division can factor N in $O(N^{1/3})$ time.[1]

Other improvements [edit]

The fundamental ideas of Fermat's factorization method are the basis of the quadratic sieve and general number field sieve, the best-known algorithms for factoring large semiprimes, which are the "worst-case". The primary improvement that quadratic sieve makes over Fermat's factorization method is that instead of simply finding a square in the sequence of $a^2 - n$, it finds a subset of elements of this sequence whose *product* is a square, and it does this in a highly efficient manner. The end result is the same: a difference of square mod *n* that, if nontrivial, can be used to factor *n*.

See also [edit]

- Completing the square
- Factorization of polynomials
- Factor theorem
- FOIL rule
- Monoid factorisation
- Pascal's triangle
- Prime factor
- Factorization
- Euler's factorization method
- Integer factorization
- Program synthesis
- Table of Gaussian integer factorizations
- Unique factorization

References [edit]

1. ^ Lehman, R. Sherman (1974). "Factoring Large Integers" (PDF). *Mathematics of Computation* **28** (126): 637–646. doi:10.2307/2005940.

- J. McKee, "Speeding Fermat's factoring method", *Mathematics of Computation*, 68:1729-1737 (1999).

External links [edit]

- Fermat's factorization running time, at blogspot.in

v · t · e	Number-theoretic algorithms [hide]
Primality tests	AKS test · APR test · Baillie–PSW · ECPP test · Elliptic curve · Pocklington · Fermat · Lucas · *Lucas–Lehmer* · *Lucas–Lehmer–Riesel* · *Proth's theorem* · *Pépin's* · Quadratic Frobenius test · Solovay–Strassen · Miller–Rabin
Prime-generating	Sieve of Atkin · Sieve of Eratosthenes · Sieve of Sundaram · Wheel factorization
Integer factorization	Continued fraction (CFRAC) · Dixon's · Lenstra elliptic curve (ECM) · Euler's · Pollard's rho · $p - 1$ · $p + 1$ · Quadratic sieve (QS) · General number field sieve (GNFS) · *Special number field sieve (SNFS)* · Rational sieve · **Fermat's** · Shanks' square forms · Trial division · Shor's
Multiplication	Ancient Egyptian · Long · Karatsuba · Toom–Cook · Schönhage–Strassen · Fürer's
Discrete logarithm	Baby-step giant-step · Pollard rho · Pollard kangaroo · Pohlig–Hellman · Index calculus · Function field sieve
Greatest common divisor	Binary · Euclidean · Extended Euclidean · Lehmer's
Modular square root	Cipolla · Pocklington's · Tonelli–Shanks
Other algorithms	Chakravala · Cornacchia · Integer relation · Integer square root · Modular exponentiation · Schoof's
	Italics indicate that algorithm is for numbers of special forms · Smallcaps indicate a deterministic algorithm

Categories: Integer factorization algorithms