Fermat’s factorization method

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.

1 The basic method

One tries various values of \(a\), hoping that \(a^2 - N = b^2\), a square.

\[
\text{FermatFactor}(N): \quad \text{// N should be odd}
\]

\[
\begin{align*}
a & \leftarrow \text{ceil}(\sqrt{N}) \\
b2 & \leftarrow a^2 - N \\
\text{while} \ b2 \text{ isn’t a square:} \\
a & \leftarrow a + 1 \quad \text{// equivalently: } b2 \\
& \leftarrow b2 + 2a + 1 \\
b2 & \leftarrow a^2 - N \quad \text{// a} \\
& \leftarrow a + 1
\end{align*}
\]

\[
\text{return a - sqrt(b2)} \quad \text{// or } a + \text{sqrt(b2)}
\]

For example, to factor \(N = 5959\), our 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.

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 \(\leq\) the square-root of \(N\). And so \(a - b = N/(a+b)\) is the largest factor \(\leq\) 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 \(c = 1\)), 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. Note that this is independent of the size of \(N\).

2 Fermat’s and trial division

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:

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:
3 Sieve improvement

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 \( N = 2345678917 \):

One can quickly tell that none of these values of \( b^2 \) 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 \( N = 2345678917 \),

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:

\[
\text{FermatSieve}(N, a_{\text{start}}, a_{\text{end}}, a_{\text{step}}, \text{modulus})
\]

\[
a \leftarrow a_{\text{start}}
\]

\[
do \text{modulus times:}
\]

\[
b^2 \leftarrow a^2 - N
\]

\[
\text{if} \quad b^2 \text{ is a square, modulo modulus:}
\]

\[
\text{FermatSieve}(N, a, a_{\text{end}}, a_{\text{step}} \ast \text{modulus}, \text{NextModulus})
\]

\[
\text{endif}
\]

\[
a \leftarrow a + a_{\text{step}}
\]

\[
enddo
\]

But the recursion is stopped when few \( a \)-values remain; that is, when \( (a_{\text{end}} - a_{\text{start}}) / a_{\text{step}} \) is small. Also, because \( a \)'s step-size is constant, one can compute successive \( b^2 \)'s with additions.

4 Multiplier improvement

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

If the approximate ratio of two factors \( \left( \frac{d}{c} \right) \) is known, then the rational number \( \frac{v}{u} \) can picked near that value. \( \frac{N_{uv}}{uv} = \frac{cv}{du} \), and the factors are roughly equal: Fermat's, applied to \( N_{uv} \), will find them quickly. Then \( \gcd(N_{uv}, cv) = c \) and \( \gcd(N_{uv}, du) = d \). (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 \( N_{uv} \). 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]

5 Other improvements

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 \( n \).

6 See also

- 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

7 References


8 External links

- Fermat's factorization running time, at blogspot.in
9 Text and image sources, contributors, and licenses

9.1 Text

- Fermat's factorization method
  
  **Contributors:** Ixfd64, Dcoetzee, Giftlite, Wroscel, Bobo192, Sligocki, TwoSchubert, Oleg Alexandrov, NeoUrfuhrner, Bubba73, Maxal, Fram, Melchoir, Eskimbot, Skintigh, Captainfranz, Patenaude, Saturday, Patrickkonor, CRGreathouse, Thjiubot, SUMpTHY, Jirka6, Magioladitis, STBotD, Alik Kirillovich, Addbot, Luckas-bot, Yobot, Xqbot, GrouchoBot, Raushc, Quondum, Fox1991, Frietjes, Helpful Pixie Bot, Mweerden, Yourskadhir, MortenEspensen, Sguv, Saevarb and Anonymous: 36

9.2 Images

9.3 Content license

- Creative Commons Attribution-Share Alike 3.0