



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Català
Deutsch
Español
Français
Italiano
עברית
Nederlands
Polski
Português
Русский
Suomi
ไทย
中文

Edit links

Create account Log in

Article **Talk**

Read **Edit** View history

Search

General number field sieve

From Wikipedia, the free encyclopedia

In **number theory**, the **general number field sieve** (**GNFS**) is the most **efficient** classical **algorithm** known for **factoring integers** larger than 100 digits. **Heuristically**, its **complexity** for factoring an integer *n* (consisting of $\lfloor \log_2 n \rfloor + 1$ bits) is of the form

$$\exp \left(\left(\sqrt[3]{\frac{64}{9}} + o(1) \right) (\ln n)^{\frac{1}{3}} (\ln \ln n)^{\frac{2}{3}} \right) = L_n \left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}} \right]$$

(in **L-notation**), where **ln** is the **natural logarithm**.^[1] It is a generalization of the **special number field sieve**: while the latter can only factor numbers of a certain special form, the general number field sieve can factor any number apart from **prime powers** (which are trivial to factor by taking roots). When the term **number field sieve** (**NFS**) is used without qualification, it refers to the general number field sieve.

The principle of the number field sieve (both special and general) can be understood as an improvement to the simpler **rational sieve** or **quadratic sieve**. When using such algorithms to factor a large number *n*, it is necessary to search for **smooth numbers** (i.e. numbers with small prime factors) of order $n^{1/2}$. The size of these values is exponential in the size of *n* (see below). The general number field sieve, on the other hand, manages to search for smooth numbers that are subexponential in the size of *n*. Since these numbers are smaller, they are more likely to be smooth than the numbers inspected in previous algorithms. This is the key to the efficiency of the number field sieve. In order to achieve this speed-up, the number field sieve has to perform computations and factorizations in **number fields**. This results in many rather complicated aspects of the algorithm, as compared to the simpler rational sieve.

Note that $\log_2 n$ is the number of bits in the binary representation of *n*, that is the size of the input to the algorithm, so any element of the order n^c for a constant *c* is exponential in $\log n$. The running time of the number field sieve is super-polynomial but sub-exponential in the size of the input.

Contents

- 1 Number fields
- 2 Method
- 3 Improving polynomial choice
- 4 Implementations
- 5 See also
- 6 Notes
- 7 References

Number fields

*Main article: **Number field***

Suppose *f* is a *k*-degree polynomial over **Q** (the rational numbers), and *r* is a complex root of *f*. Then, *f*(*r*) = 0, which can be rearranged to express r^k as a linear combination of powers of *r* less than *k*. This equation can be used to reduce away any powers of $r \geq k$. For example, if *f*(*x*) = *x*² + 1 and *r* is the imaginary unit *i*, then *i*² + 1=0, or *i*² = −1. This allows us to define the complex product:

$$(a + bi)(c + di) = ac + (ad + bc)i + (bd)i^2 = (ac - bd) + (ad + bc)i.$$

In general, this leads directly to the **algebraic number field** **Q**[*r*], which can be defined as the set of real numbers given by:

$$a_{k-1}r^{k-1} + \ldots + a_1r^1 + a_0r^0, \text{ where } a_0, \ldots, a_{k-1} \text{ in } \mathbf{Q}.$$

The product of any two such values can be computed by taking the product as polynomials, then reducing any powers of $r \geq k$ as described above, yielding a value in the same form. To ensure that this field is actually *k*-dimensional and does not collapse to an even smaller field, it is sufficient that *f* is an **irreducible polynomial** over the rationals. Similarly, one may define the **ring of integers** **O**_{**Q**[*r*]} as the subset of **Q**[*r*] where *a*₀,...,*a*_{*k*−1} are restricted to be integers. In some cases, this ring of integers is equivalent to the ring **Z**[*r*]. However, there are

many exceptions, such as for $\mathbf{Q}[\sqrt{d}]$ when d is equal to 1 modulo 4.^[2]

Method ^[edit]

Two [polynomials](#) $f(x)$ and $g(x)$ of small [degrees](#) d and e are chosen, which have integer coefficients, which are [irreducible](#) over the [rationals](#), and which, when interpreted [mod \$n\$](#) , have a common integer [root](#) m . An optimal strategy for choosing these polynomials is not known; one simple method is to pick a degree d for a polynomial, consider the expansion of n in [base \$m\$](#) (allowing digits between $-m$ and m) for a number of different m of order $n^{1/d}$, and pick $f(x)$ as the polynomial with the smallest coefficients and $g(x)$ as $x - m$.

Consider the number field rings $\mathbf{Z}[r_1]$ and $\mathbf{Z}[r_2]$, where r_1 and r_2 are roots of the polynomials f and g . Since f is of degree d with integer coefficients, if a and b are integers, then so will be $b^d \cdot f(a/b)$, which we call r . Similarly, $s = b^e \cdot g(a/b)$ is an integer. The goal is to find integer values of a and b that simultaneously make r and s [smooth](#) relative to the chosen basis of primes. If a and b are small, then r and s will be small too, about the size of m , and we have a better chance for them to be smooth at the same time. The current best-known approach for this search is [lattice sieving](#); to get acceptable yields, it is necessary to use a large factor base.

Having enough such pairs, using [Gaussian elimination](#), one can get products of certain r and of the corresponding s to be squares at the same time. A slightly stronger condition is needed—that they are [norms](#) of squares in our number fields, but that condition can be achieved by this method too. Each r is a norm of $a - r_1 b$ and hence that the product of the corresponding factors $a - r_1 b$ is a square in $\mathbf{Z}[r_1]$, with a "square root" which can be determined (as a product of known factors in $\mathbf{Z}[r_1]$)—it will typically be represented as an irrational [algebraic number](#). Similarly, the product of the factors $a - r_2 b$ is a square in $\mathbf{Z}[r_2]$, with a "square root" which also can be computed. It should be remarked that the use of Gaussian elimination does not give the optimal run time of the algorithm. Instead, sparse matrix solving algorithms such as [Block Lanczos](#) or [Block Wiedemann](#) are used.

Since m is a root of both f and g mod n , there are [homomorphisms](#) from the rings $\mathbf{Z}[r_1]$ and $\mathbf{Z}[r_2]$ to the ring $\mathbf{Z}/n\mathbf{Z}$ (the integers [mod \$n\$](#)), which map r_1 and r_2 to m , and these homomorphisms will map each "square root" (typically not represented as a rational number) into its integer representative. Now the product of the factors $a - mb$ mod n can be obtained as a square in two ways—one for each homomorphism. Thus, one can find two numbers x and y , with $x^2 - y^2$ divisible by n and again with probability at least one half we get a factor of n by finding the [greatest common divisor](#) of n and $x - y$.

Improving polynomial choice ^[edit]

The choice of polynomial can dramatically affect the time to complete the remainder of the algorithm. The method of choosing polynomials based on the expansion of n in base m shown above is suboptimal in many practical situations, leading to the development of better methods.

One such method was suggested by Murphy and Brent;^[3] they introduce a two-part score for polynomials, based on the presence of roots modulo small primes and on the average value that the polynomial takes over the sieving area.

The best reported results^[4] were achieved by the method of [Thorsten Kleinjung](#),^[5] which allows $g(x) = ax + b$, and searches over a composed of small prime factors congruent to 1 modulo $2d$ and over leading coefficients of f which are divisible by 60.

Implementations ^[edit]

Some implementations focus on a certain smaller class of numbers. These are known as [special number field sieve](#) techniques, such as used in the [Cunningham project](#). A project called NFSNET ran from 2002^[6] through at least 2007. It used volunteer distributed computing on the [Internet](#).^[7] [Paul Leyland](#) of the [United Kingdom](#) and Richard Wackerbarth of Texas were involved.^[8]

Until 2007, the gold-standard implementation was a suite of software developed and distributed by [CWI](#) in the Netherlands, which was available only under a relatively restrictive license. In 2007, [Jason Papadopoulos](#) developed a faster implementation of final processing as part of msieve, which is public-domain. Both implementations feature the ability to be distributed among several nodes in a cluster with a sufficiently fast interconnect.

Polynomial selection is normally performed by [GPL](#) software written by Kleinjung, or by msieve, and lattice sieving by GPL software written by Franke and Kleinjung; these are distributed in GGNFS.

- [NFS@Home](#) ^[↗]
- [GGNFS](#) ^[↗]

- [pGNFS](#)
- [factor by gnfs](#)
- [CADO-NFS](#)
- [msieve](#), which contains excellent final-processing code, a good implementation of the polynomial selection which is very good for smaller numbers, and an implementation of the line sieve.
- [kmGNFS](#)

See also [\[edit\]](#)

- [Special number field sieve](#)

Notes [\[edit\]](#)

1.

^

Pomerance, Carl (December 1996). "A Tale of Two Sieves" (PDF). *Notices of the AMS* **43** (12). pp. 1473–1485.

2.

^

Ribenboim, Paulo (1972). *Algebraic Numbers*. Wiley-Interscience. ISBN 0-471-71804-1.

3.

^

Murphy, B.; Brent, R. P. (1998), "On quadratic polynomials for the number field sieve" , *Australian Computer Science Communications* **20**: 199–213

4.

^

Franke, Jens (2006), *On RSA 200 and larger projects* (PDF)

5.

^

Kleinjung, Thorsten (October 2006). "On polynomial selection for the general number field sieve" (PDF). *Mathematics of Computation* **75** (256): 2037–2047. doi:10.1090/S0025-5718-06-01870-9. Retrieved 2007-12-13.

6.

^

Paul Leyland (December 12, 2003). "NFSNET: the first year" . *Presentation at EIDMA-CWI Workshop on Factoring Large Numbers*. Retrieved August 9, 2011.

7.

^

"Welcome to NFSNET" . April 23, 2007. Archived from the original on October 22, 2007. Retrieved August 9, 2011.

8.

^

"About NFSNET" . Archived from the original on May 9, 2008. Retrieved August 9, 2011.

References [\[edit\]](#)

•

Arjen K. Lenstra and H. W. Lenstra, Jr. (eds.). "The development of the number field sieve". *Lecture Notes in Math.* (1993) 1554. Springer-Verlag.

•

Richard Crandall and Carl Pomerance. *Prime Numbers: A Computational Perspective* (2001). 2nd edition, Springer. ISBN 0-387-25282-7. Section 6.2: Number field sieve, pp. 278–301.

<div><div><div>V</div><div>•</div><div>T</div><div>•</div><div>E</div></div></div>	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 · <i>p</i> − 1 · <i>p</i> + 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	
<i>Italics indicate that algorithm is for numbers of special forms · Smallcaps indicate a deterministic algorithm</i>		

Categories: [Integer factorization algorithms](#)

This page was last modified on 20 June 2015, at 21:26.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#)


[About Wikipedia](#)


[Disclaimers](#)

[Contact Wikipedia](#)

[Developers](#)

[Mobile view](#)

a WIKIMEDIA project

Powered By MediaWiki