Integer factorization

“Prime decomposition” redirects here. For the prime decomposition theorem for 3-manifolds, see Prime decomposition (3-manifold).

In number theory, **integer factorization** is the decomposition of a composite number into a product of smaller integers. If these integers are further restricted to prime numbers, the process is called **prime factorization**.

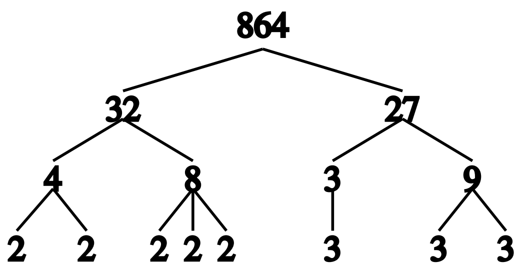
When the numbers are very large, no efficient, non-quantum integer factorization algorithm is known. An effort by several researchers, concluded in 2009, to factor a 232-digit number (RSA-768) utilizing hundreds of machines took two years and the researchers estimated that a 1024-bit RSA modulus would take about a thousand times as long.[1] However, it has not been proven that no efficient algorithm exists. The presumed difficulty of this problem is at the heart of widely used algorithms in cryptography such as RSA. Many areas of mathematics and computer science have been brought to bear on the problem, including elliptic curves, algebraic number theory, and quantum computing.

Not all numbers of a given length are equally hard to factor. The hardest instances of these problems (for currently known techniques) are semiprimes, the product of two prime numbers. When they are both large, for instance more than two thousand bits long, randomly chosen, and about the same size (but not too close, e.g., to avoid efficient factorization by Fermat’s factorization method), even the fastest prime factorization algorithms on the fastest computers can take enough time to make the search impractical; that is, as the number of digits of the primes being factored increases, the number of operations required to perform the factorization on any computer increases drastically.

Many cryptographic protocols are based on the difficulty of factoring large composite integers or a related problem—for example, the RSA problem. An algorithm that efficiently factors an arbitrary integer would render RSA-based public-key cryptography insecure.

1 Prime decomposition

By the fundamental theorem of arithmetic, every positive integer has a unique prime factorization. (By convention 1 is the empty product.) If the integer is prime then it can be recognized as such in polynomial time. If composite however, the theorem gives no insight into how to obtain the factors.



This image demonstrates the prime decomposition of 864. A shorthand way of writing the resulting prime factors is $2^5 \times 3^3$

Given a general algorithm for integer factorization, any integer can be factored down to its constituent prime factors simply by repeated application of this algorithm. The situation is more complicated with special-purpose factorization algorithms, whose benefits may not be realized as well or even at all with the factors produced during decomposition. For example, if $N = 10 \times p \times q$ where $p < q$ are very large primes, trial division will quickly produce the factors 2 and 5 but will take p divisions to find the next factor. As a contrasting example, if N is the product of the primes 13729, 1372933, and 18848997161, where $13729 \times 1372933 = 18848997157$, Fermat’s factorization method will start out with $a = \lceil\sqrt{N}\rceil = 18848997159$ which immediately yields $b = \sqrt{a^2 - N} = \sqrt{4} = 2$ and hence the factors $a - b = 18848997157$ and $a + b = 18848997161$. While these are easily recognized as respectively composite and prime, Fermat’s method will take much longer to factorize the composite one because the starting value of $\lceil\sqrt{18848997157}\rceil = 137292$ for a is nowhere near 1372933.

2 Current state of the art

See also: integer factorization records

Among the b-bit numbers, the most difficult to factor in practice using existing algorithms are those that are products of two primes of similar size. For this reason, these are the integers used in cryptographic applications. The largest such semiprime yet factored was RSA-768, a 768-bit number with 232 decimal digits, on December 12, 2009.[1] This factorization was a collaboration of several research institutions, spanning two years and taking the equivalent of almost 2000 years of computing on a single-core 2.2 GHz AMD Opteron. Like all recent factorization records, this factorization was completed with a highly optimized implementation of the general number field sieve run on hundreds of machines.

2.1 Difficulty and complexity

No algorithm has been published that can factor all integers in polynomial time, i.e., that can factor b-bit numbers in time $O(b^k)$ for some constant k. Neither the existence nor non-existence of such algorithms has been proved, but it is generally suspected that they do not exist and hence that the problem is not in class P. The problem is clearly in class NP but has not been proved to be or not be NP-complete. It is generally suspected not to be NP-complete.

There are published algorithms that are faster than $O((1+\varepsilon)^b)$ for all positive ε, i.e., sub-exponential. The best published asymptotic running time is for the general number field sieve (GNFS) algorithm, which, for a b-bit number n, is:

$$O\left(\exp \sqrt[3]{\frac{64}{9} b(\log b)^2}\right).$$

For current computers, GNFS is the best published algorithm for large n (more than about 100 digits). For a quantum computer, however, Peter Shor discovered an algorithm in 1994 that solves it in polynomial time. This will have significant implications for cryptography if quantum computation is possible. Shor's algorithm takes only $O(b^3)$ time and $O(b)$ space on b-bit number inputs. In 2001, the first seven-qubit quantum computer became the first to run Shor's algorithm. It factored the number 15.[2]

When discussing what complexity classes the integer factorization problem falls into, it is necessary to distinguish two slightly different versions of the problem:

- The function problem version: given an integer N, find an integer d with $1 < d < N$ that divides N (or conclude that N is prime). This problem is trivially in FNP and it's not known whether it lies in FP or not. This is the version solved by practical implementations.

- The decision problem version: given an integer N and an integer M with $1 < M < N$, does N have a factor d with $1 < d \leq M$? This version is useful because most well studied complexity classes are defined as classes of decision problems, not function problems.

For $\sqrt{N} \leq M < N$, the decision problem is equivalent to asking if N is not prime.

An algorithm for either version provides one for the other. Repeated application of the function problem (applied to d and N/d, and their factors, if needed) will eventually provide either a factor of N no larger than M or a factorization into primes all greater than M. All known algorithms for the decision problem work in this way. Hence it is only of theoretical interest that, with at most $\log N$ queries using an algorithm for the decision problem, one would isolate a factor of N (or prove it prime) by binary search.

It is not known exactly which complexity classes contain the decision version of the integer factorization problem. It is known to be in both NP and co-NP. This is because both YES and NO answers can be verified in polynomial time. An answer of YES can be certified by exhibiting a factorization $N = d(N/d)$ with $d \leq M$. An answer of NO can be certified by exhibiting the factorization of N into distinct primes, all larger than M. We can verify their primality using the AKS primality test and that their product is N by multiplication. The fundamental theorem of arithmetic guarantees that there is only one possible string that will be accepted (providing the factors are required to be listed in order), which shows that the problem is in both **UP** and **co-UP**.[3] It is known to be in BQP because of Shor's algorithm. It is suspected to be outside of all three of the complexity classes P, NP-complete, and co-NP-complete. It is therefore a candidate for the NP-intermediate complexity class. If it could be proved that it is in either NP-Complete or co-NP-Complete, that would imply NP = co-NP. That would be a very surprising result, and therefore integer factorization is widely suspected to be outside both of those classes. Many people have tried to find classical polynomial-time algorithms for it and failed, and therefore it is widely suspected to be outside P.

In contrast, the decision problem "is N a composite number?" (or equivalently: "is N a prime number?") appears to be much easier than the problem of actually finding the factors of N. Specifically, the former can be solved in polynomial time (in the number n of digits of N) with the AKS primality test. In addition, there are a number of probabilistic algorithms that can test primality very quickly in practice if one is willing to accept the vanishingly small possibility of error. The ease of primality testing is a crucial part of the RSA algorithm, as it is necessary to find large prime numbers to start with.

3 Factoring algorithms

3.1 Special-purpose

A special-purpose factoring algorithm's running time depends on the properties of the number to be factored or on one of its unknown factors: size, special form, etc. Exactly what the running time depends on varies between algorithms.

An important subclass of special-purpose factoring algorithms is the *Category 1* or *First Category* algorithms, whose running time depends on the size of smallest prime factor. Given an integer of unknown form, these methods are usually applied before general-purpose methods to remove small factors.[4] For example, trial division is a Category 1 algorithm.

- Trial division
- Wheel factorization
- Pollard's rho algorithm
- Algebraic-group factorisation algorithms, among which are Pollard's $p-1$ algorithm, Williams' $p+1$ algorithm, and Lenstra elliptic curve factorization
- Fermat's factorization method
- Euler's factorization method
- Special number field sieve

3.2 General-purpose

A general-purpose factoring algorithm, also known as a *Category 2*, *Second Category*, or *Kraitchik family* algorithm (after Maurice Kraitchik),[4] has a running time which depends solely on the size of the integer to be factored. This is the type of algorithm used to factor RSA numbers. Most general-purpose factoring algorithms are based on the congruence of squares method.

- Dixon's algorithm
- Continued fraction factorization (CFRAC)
- Quadratic sieve
- Rational sieve
- General number field sieve
- Shanks' square forms factorization (SQUFOF)

3.3 Other notable algorithms

- Shor's algorithm, for quantum computers

4 Heuristic running time

In number theory, there are many integer factoring algorithms that heuristically have expected running time

$$L_n\left[\tfrac{1}{2}, 1+o(1)\right] = e^{(1+o(1))\sqrt{(\log n)(\log\log n)}}$$

in big O and L-notation. Some examples of those algorithms are the elliptic curve method and the quadratic sieve. Another such algorithm is the **class group relations method** proposed by Schnorr,[5] Seysen,[6] and Lenstra,[7] that is proved under the assumption of the Generalized Riemann Hypothesis (GRH).

5 Rigorous running time

The Schnorr-Seysen-Lenstra probabilistic algorithm has been rigorously proven by Lenstra and Pomerance[8] to have expected running time $L_n\left[\frac{1}{2}, 1+o(1)\right]$ by replacing the GRH assumption with the use of multipliers. The algorithm uses the class group of positive binary quadratic forms of discriminant Δ denoted by $G\Delta$. $G\Delta$ is the set of triples of integers (a, b, c) in which those integers are relative prime.

5.1 Schnorr-Seysen-Lenstra Algorithm

Given is an integer n that will be factored, where n is an odd positive integer greater than a certain constant. In this factoring algorithm the discriminant Δ is chosen as a multiple of n, $\Delta = -dn$, where d is some positive multiplier. The algorithm expects that for one d there exist enough smooth forms in $G\Delta$. Lenstra and Pomerance show that the choice of d can be restricted to a small set to guarantee the smoothness result.

Denote by $P\Delta$ the set of all primes q with Kronecker symbol $\left(\frac{\Delta}{q}\right) = 1$. By constructing a set of generators of $G\Delta$ and prime forms f_q of $G\Delta$ with q in $P\Delta$ a sequence of relations between the set of generators and f_q are produced. The size of q can be bounded by $c_0(\log|\Delta|)^2$ for some constant c_0.

The relation that will be used is a relation between the product of powers that is equal to the neutral element of $G\Delta$. These relations will be used to construct a so-called ambiguous form of $G\Delta$, which is an element of $G\Delta$ of order dividing 2. By calculating the corresponding factorization of Δ and by taking a gcd, this ambiguous form provides the complete prime factorization of n. This algorithm has these main steps:

Let n be the number to be factored.

1. Let Δ be a negative integer with $\Delta = -dn$, where d is a multiplier and Δ is the negative discriminant of some quadratic form.
2. Take the t first primes $p_1 = 2, p_2 = 3, p_3 = 5, \dots, p_t$, for some $t \in \mathbb{N}$.
3. Let f_q be a random prime form of $G\Delta$ with $\left(\frac{\Delta}{q}\right) = 1$.
4. Find a generating set X of $G\Delta$

5. Collect a sequence of relations between set X and $\{fq : q \in P\Delta\}$ satisfying: $\left(\prod_{x \in X} x^{r(x)}\right) \cdot \left(\prod_{q \in P_\Delta} f_q^{t(q)}\right) = 1$

6. Construct an ambiguous form that is an element $f \in G\Delta$ of order dividing 2 to obtain a coprime factorization of the largest odd divisor of Δ in which $\Delta = -4ac$ or $a(a-4c)$ or $(b-2a)(b+2a)$

7. If the ambiguous form provides a factorization of n then stop, otherwise find another ambiguous form until the factorization of n is found. In order to prevent useless ambiguous forms from generating, build up the 2-Sylow group $\mathrm{Sll}_2(\Delta)$ of $G(\Delta)$.

To obtain an algorithm for factoring any positive integer, it is necessary to add a few steps to this algorithm such as trial division, and the Jacobi sum test.

5.2 Expected running time

The algorithm as stated is a probabilistic algorithm as it makes random choices. Its expected running time is at most $L_n\left[\frac{1}{2}, 1+o(1)\right]$.[8]

6 See also

- Canonical representation of a positive integer
- Factorization
- Multiplicative partition
- Partition (number theory) – a way of writing a number as a sum of positive integers.

7 Notes

[1] Kleinjung; et al. (2010-02-18). "Factorization of a 768-bit RSA modulus" (PDF). International Association for Cryptologic Research. Retrieved 2010-08-09.

[2] Lieven M. K., Vandersypen; et al. (2007-12-27). "NMR quantum computing: Realizing Shor's algorithm". Nature. Retrieved 2010-08-09.

[3] Lance Fortnow (2002-09-13). "Computational Complexity Blog: Complexity Class of the Week: Factoring".

[4] David Bressoud and Stan Wagon (2000). *A Course in Computational Number Theory*. Key College Publishing/Springer. pp. 168–69. ISBN 978-1-930190-10-8.

[5] Schnorr, Claus P. (1982). "Refined analysis and improvements on some factoring algorithms". *Journal of Algorithms*. **3** (2): 101–127. doi:10.1016/0196-6774(82)90012-8. MR 0657269.

[6] Seysen, Martin (1987). "A probabilistic factorization algorithm with quadratic forms of negative discriminant". *Mathematics of Computation*. **48** (178): 757–780. doi:10.1090/S0025-5718-1987-0878705-X. MR 0878705.

[7] Lenstra, Arjen K (1988). "Fast and rigorous factorization under the generalized Riemann hypothesis". *Indagationes Mathematicae*. **50**: 443–454.

[8] Lenstra, H. W.; Pomerance, Carl (July 1992). "A Rigorous Time Bound for Factoring Integers" (PDF). *Journal of the American Mathematical Society*. **5** (3): 483–516. doi:10.1090/S0894-0347-1992-1137100-0. MR 1137100.

8 References

- Richard Crandall and Carl Pomerance (2001). *Prime Numbers: A Computational Perspective*. Springer. ISBN 0-387-94777-9. Chapter 5: Exponential Factoring Algorithms, pp. 191–226. Chapter 6: Subexponential Factoring Algorithms, pp. 227–284. Section 7.4: Elliptic curve method, pp. 301–313.
- Donald Knuth. *The Art of Computer Programming*, Volume 2: *Seminumerical Algorithms*, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89684-2. Section 4.5.4: Factoring into Primes, pp. 379–417.

9 External links

- msieve - SIQS and NFS - has helped complete some of the largest public factorizations known
- Richard P. Brent, "Recent Progress and Prospects for Integer Factorisation Algorithms", *Computing and Combinatorics"*, 2000, pp. 3–22. download
- Manindra Agrawal, Neeraj Kayal, Nitin Saxena, "PRIMES is in P." Annals of Mathematics 160(2): 781-793 (2004). August 2005 version PDF
- Eric W. Weisstein, "RSA-640 Factored" *MathWorld Headline News*, November 8, 2005

10 Text and image sources, contributors, and licenses

10.1 Text

- **Integer factorization** *Source:* https://en.wikipedia.org/wiki/Integer_factorization?oldid=775148465 *Contributors:* AxelBoldt, LC~enwiki, The Anome, XJaM, Roadrunner, Cwitty, RTC, Michael Hardy, Palnatoke, Nixdorf, Shellreef, Ixfd64, TakuyaMurata, GTBacchus, Snoyes, Julesd, Ciphergoth, Charles Matthews, Dcoetzee, Doradus, Birkett, Itai, Jimbreed, Fredrik, Pingveno, Henrygb, Decrypt3, Giftlite, Herbee, Tromer, Gracefool, Macrakis, Matt Crypto, Deego, Gdr, Sonjaaa, Wroscel, Creidieki, Petershank, Heryu~enwiki, Rich Farmbrough, Guanabot, Bender235, Bobo192, Alansohn, Craigy144, Sligocki, Fadereu, Oleg Alexandrov, SteveHFish, Tbsmith, OwenX, Linas, Mindmatrix, Shreevatsa, MamaGeek, Ruud Koot, Eclecticos, Jeff3000, Wandadubbayou, MFH, Dionyziz, Pinkgothic, Abd, Graham87, Rjwilmsi, JVz, Kinu, Bubba73, Fivemack, Maxal, Glenn L, Chobot, Jersey Devil, Siddhant, Wavelength, Xcelerate, Gene.arboit, RussBot, Taejo, Froth, Mysid, Klutzy, Cedar101, KGasso, Claygate, Vicarious, Gesslein, RunOrDie, Nahaj, SmackBot, RDBury, Maksim-e~enwiki, Stux, Monz, Rōnin, Gilliam, Jushi, Chris the speller, Tbandrow, Skintigh, MrDrBob, Agateller, Jprg1966, PrimeHunter, Pimantony, Gutworth, Cophus, Cybercobra, DRLB, Drphilharmonic, Andrei Stroe, Autopilot, Lambiam, Trou, IronGargoyle, Luokehao, Mets501, Stephen B Streater, Vaughan Pratt, CRGreathouse, Wafulz, Qc, HenningThielemann, GrGr, Tkircher, Schneau, Marek69, Escarbot, Markulf, ForrestVoight, VectorPosse, Britcruise, Serpent's Choice, JAnDbot, Starrynte, Tim333, VoABot II, Baccyak4H, Doranb, A3nm, David Eppstein, DerHexer, R'n'B, Uncle Dick, Derlay, Lupussy, Tarotcards, R00723r0, ABF, Gentlemath, Nxavar, Jimbo Grilles, Garde, Branger~enwiki, Faradayplank, Skippydo, Lordfeepness, Anchor Link Bot, DRTllbrg, ClueBot, Trojancowboy, Justin W Smith, Fyyer, Niceguyedc, FromUA, JNLII, Bywater100, Dekart, Roentgenium111, MrOllie, LaaknorBot, Glane23, Tide rolls, דניאל ב., Bpapa2, AnomieBOT, Dr0nnl, Citation bot, Jeffrey Mall, Cyndaquazy, Jonas AGX, Teslavitas, DefenseSupportParty, Raulshc, TLange, Velociostrich, XxTimberlakexx, 10metreh, Shelbymoore3, Jel3j2, E.songhori, RjwilmsiBot, Igor Yalovecky, EmausBot, WikitanvirBot, Stringfoto, Bdijkstra, Lildudeman98, Starlet9, ChuispastonBot, Любослов Езыкин, Anita5192, Buhrow52, ClueBot NG, Wcherowi, Jmstacey, ElisabetFatimatulVeronika, Rezabot, MerlIwBot, Helpful Pixie Bot, Delerium2k12, Wikisian, Dan653, MC-CPO, Iczero, ChrisGualtieri, Jochen Burghardt, FenixFeather, CsDix, Kolophon, MathLine, JustBerry, JaconaFrere, Monkbot, Eurodyne, Nerdgirl22, DeityPsymon, Hatemisleadingwords and Anonymous: 191

10.2 Images

- **File:Lattice_of_the_divisibility_of_60.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/51/Lattice_of_the_divisibility_of_60.svg *License:* CC-BY-SA-3.0 *Contributors:* No machine-readable source provided. Own work assumed (based on copyright claims). *Original artist:* No machine-readable author provided. Ed g2s assumed (based on copyright claims).
- **File:PrimeDecompositionExample.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/bf/PrimeDecompositionExample.svg *License:* CC0 *Contributors:* Own work *Original artist:* Krishnavedala
- **File:Question_dropshade.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/dd/Question_dropshade.png *License:* Public domain *Contributors:* Image created by JRM *Original artist:* JRM

10.3 Content license