Shor's algorithm

From Wikipedia, the free encyclopedia

Shor's algorithm, named after mathematician Peter Shor, is a quantum algorithm (an algorithm that runs on a quantum computer) for integer factorization formulated in 1994. Informally it solves the following problem: given an integer *N*, find its prime factors.

On a quantum computer, to factor an integer *N*, Shor's algorithm runs in polynomial time (the time taken is polynomial in log *N*, which is the size of the input).[1] Specifically it takes time and quantum gates of order $O((\log N)^2(\log\log N)(\log\log\log N))$ using fast multiplication,[2] demonstrating that the integer factorization problem can be efficiently solved on a quantum computer and is thus in the complexity class **BQP**. This is substantially faster than the most efficient known classical factoring algorithm, the general number field sieve, which works in sub-exponential time — about $O(e^{1.9\,(\log N)^{1/3}\,(\log\log N)^{2/3}})$.[3] The efficiency of Shor's algorithm is due to the efficiency of the quantum Fourier transform, and modular exponentiation by repeated squarings.

If a quantum computer with a sufficient number of qubits could operate without succumbing to noise and other quantum decoherence phenomena, Shor's algorithm could be used to break public-key cryptography schemes such as the widely used RSA scheme. RSA is based on the assumption that factoring large numbers is computationally intractable. So far as is known, this assumption is valid for classical (non-quantum) computers; no classical algorithm is known that can factor in polynomial time. However, Shor's algorithm shows that factoring is efficient on an ideal quantum computer, so it may be feasible to defeat RSA by constructing a large quantum computer. It was also a powerful motivator for the design and construction of quantum computers and for the study of new quantum computer algorithms. It has also facilitated research on new cryptosystems that are secure from quantum computers, collectively called post-quantum cryptography.

In 2001, Shor's algorithm was demonstrated by a group at IBM, who factored 15 into 3 × 5, using an NMR implementation of a quantum computer with 7 qubits.[4] After IBM's implementation, two independent groups, one at the University of Science and Technology of China, and the other one at the University of Queensland, have implemented Shor's algorithm using photonic qubits, emphasizing that multi-qubit entanglement was observed when running the Shor's algorithm circuits.[5][6] In 2012, the factorization of 15 was repeated.[7] Also in 2012, the factorization of 21 was achieved, setting the record for the largest number factored with a quantum computer.[8] In April 2012, the factorization of 143 was achieved, although this used adiabatic quantum computation rather than Shor's algorithm.[9] It was discovered in November 2014 that this adiabatic quantum computation in 2012 had in fact also factored larger numbers, the largest being 56153, which is currently the record for the largest integer factored on a quantum device.[10][11]

Contents [hide]

Procedure [edit]

The problem we are trying to solve is: given an odd composite number N, find an integer d, strictly between 1 and N, that divides N. We are interested in odd values of N because any even value of N trivially has the number 2 as a prime factor. We can use a primality testing algorithm to make sure that N is indeed composite.

Moreover, for the algorithm to work, we need N not to be the power of a prime. This can be tested by taking square, cubic, ..., k-roots of N, for $k \leq \log_2(N)$, and checking that none of these is an integer. (This

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
العربية
Català
Deutsch
Español
فارسی
Français
한국어
Italiano
עברית
Lietuvių
Magyar
Nederlands
Polski
Português
Русский
Српски / srpski
Srpskohrvatski / српскохрватски
Suomi
ไทย
Türkçe
中文
Edit links

actually excludes that $N = M^k$ for some integer M and $k > 1$.)

Since N is not a power of a prime, it is the product of two coprime numbers greater than 1. As a consequence of the Chinese remainder theorem, the number 1 has at least four distinct square roots modulo N, two of them being 1 and -1. The aim of the algorithm is to find a square root b of one, other than 1 and -1; such a b will lead to a factorization of N, as in other factoring algorithms like the quadratic sieve.

In turn, finding such a b is reduced to finding an element a of even period with a certain additional property (as explained below, it is required that the condition of Step 6 of the classical part does not hold). The quantum algorithm is used for finding the period of randomly chosen elements a, as order-finding is a hard problem on a classical computer.

Shor's algorithm consists of two parts:

1. A reduction, which can be done on a classical computer, of the factoring problem to the problem of order-finding.
2. A quantum algorithm to solve the order-finding problem.

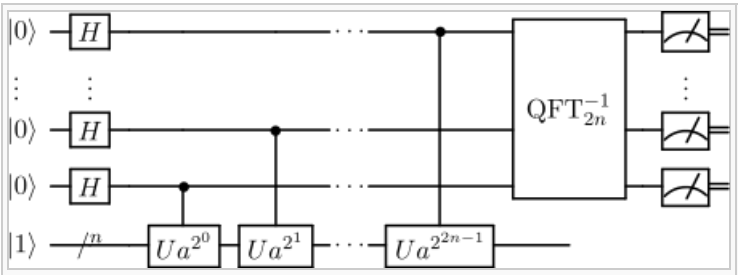
Classical part [edit]

1. Pick a random number *a* < *N*.
2. Compute gcd(*a*, *N*). This may be done using the Euclidean algorithm.
3. If gcd(*a*, *N*) ≠ 1, then it is a nontrivial factor of *N*, so we are done.
4. Otherwise, use the period-finding subroutine (below) to find *r*, the period of the following function:
   $$f(x) = a^x \bmod N,$$
   i.e. the order r of a in $(\mathbb{Z}_N)^\times$, which is the smallest positive integer *r* for which $f(x+r) = f(x)$, or $f(x+r) = a^{x+r} \bmod N = a^x \bmod N.$
5. If *r* is odd, go back to step 1.
6. If $a^{r/2} \equiv -1 \pmod N$, go back to step 1.
7. gcd($a^{r/2} + 1$, *N*) and gcd($a^{r/2} - 1$, *N*) are both nontrivial factors of *N*. We are done.

For example: $N = 15, a = 7, r = 4$, $gcd(7^2 \pm 1, 15) = gcd(49 \pm 1, 15)$, where $gcd(48, 15) = 3$ and $gcd(50, 15) = 5$.

Quantum part: Period-finding subroutine [edit]

This section **may be too technical for most readers to understand**. Please help improve this section to make it understandable to non-experts, without removing the technical details. The talk page may contain suggestions. *(February 2014)*



Quantum subroutine in Shor's algorithm.

The quantum circuits used for this algorithm are custom designed for each choice of *N* and each choice of the random *a* used in $f(x) = a^x \bmod N$. Given *N*, find $Q = 2^q$ such that $N^2 \le Q < 2N^2$, which implies $Q/r > N$. The input and output qubit registers need to hold superpositions of values from 0 to $Q - 1$, and so have *q* qubits each. Using what might appear to be twice as many qubits as necessary guarantees that there are at least *N* different *x* which produce the same *f*(*x*), even as the period *r* approaches *N*/2.

Proceed as follows:

1. Initialize the registers to
   $$Q^{-1/2} \sum_{x=0}^{Q-1} |x\rangle\,|0\rangle$$
   where *x* runs from 0 to $Q - 1$. This initial state is a superposition of *Q* states.
2. Construct *f*(*x*) as a quantum function and apply it to the above state, to obtain
   $$Q^{-1/2} \sum_x |x, f(x)\rangle\,.$$
   This is still a superposition of *Q* states.

3. Apply the quantum Fourier transform to the input register. This transform (operating on a superposition of power-of-two $Q = 2^q$ states) uses a Q^{th} root of unity such as $\omega = e^{2\pi i/Q}$ to distribute the amplitude of any given $|x\rangle$ state equally among all Q of the $|y\rangle$ states, and to do so in a different way for each different x.
   - Let y be one of the r possible integers modulo N such that yr/Q is an integer; then
   $$U_{QFT}\,|x\rangle = Q^{-1/2}\sum_y \omega^{xy}\,|y\rangle\,.$$
   This leads to the final state
   $$Q^{-1}\sum_x\sum_y \omega^{xy}\,|y, f(x)\rangle\,.$$
   Now we reorder this sum as
   $$Q^{-1}\sum_z\sum_y |y, z\rangle \sum_{x:\,f(x)=z} \omega^{xy}.$$
   This is a superposition of many more than Q states, but many fewer than Q^2 states, since there are fewer than Q distinct values of $z = f(x)$. Let
   - $\omega = e^{2\pi i/Q}$ be a Q^{th} root of unity,
   - r be the period of f,
   - x_0 be the smallest of the x which have $f(x) = z$ (we have $x_0 < r$), and
   - b indexes these x, running from 0 to $\lfloor (Q - x_0 - 1)/r \rfloor$ so that $x_0 + rb < Q$.

   Then ω^{ry} is a unit vector in the complex plane (ω is a root of unity and r and y are integers), and the coefficient of $Q^{-1}\,|y, z\rangle$ in the final state is
   $$\sum_{x:\,f(x)=z} \omega^{xy} = \sum_b \omega^{(x_0+rb)y} = \omega^{x_0 y}\sum_b \omega^{rby}.$$
   Each term in this sum represents a *different path to the same result*, and quantum interference occurs—constructive when the unit vectors ω^{ryb} point in nearly the same direction in the complex plane, which requires that ω^{ry} point along the positive real axis.
4. Perform a measurement. We obtain some outcome y in the input register and z in the output register. Since f is periodic, the probability of measuring some pair y and z is given by
   $$\left|Q^{-1}\sum_{x:\,f(x)=z} \omega^{xy}\right|^2 = Q^{-2}\left|\sum_b \omega^{(x_0+rb)y}\right|^2 = Q^{-2}\left|\sum_b \omega^{bry}\right|^2.$$
   Analysis now shows that this probability is higher the closer the unit vector ω^{ry} is to the positive real axis, or the closer yr/Q is to an integer. Unless r is a power of 2, it won't be a factor of Q.
5. Since yr/Q is close to some integer c, the known value y/Q is close to the unknown value c/r. Performing [classical] continued fraction expansion on y/Q allows us to find approximations d/s of it which satisfy two conditions:
   - A: s<N
   - B: |y/Q - d/s| < 1/2Q.

   Given these conditions (and assuming d/s is irreducible), s is very likely to be the appropriate period r, or at least a factor of it.
6. Check [classically] if $f(x) = f(x+s) \Leftrightarrow a^r \equiv 1 \pmod N$ If so, we are done.
7. Otherwise, [classically] obtain more candidates for r by using multiples of s, or by using other s with d/s near y/Q. If any candidate works, we are done.
8. Otherwise, try again starting from step 1 of this subroutine.

Explanation of the algorithm [edit]

The algorithm is composed of two parts. The first part of the algorithm turns the factoring problem into the problem of finding the period of a function, and may be implemented classically. The second part finds the period using the quantum Fourier transform, and is responsible for the quantum speedup.

Obtaining factors from period [edit]

The integers less than N and coprime with N form a finite abelian group G under multiplication modulo N. The size is given by Euler's totient function $\phi(N)$. By the end of step 3, we have an integer a in this group. Since

the group is finite, *a* must have a finite order *r*, the smallest positive integer such that

$$a^r \equiv 1 \bmod N.$$

Therefore, *N* divides (also written |) $a^r - 1$. Suppose we are able to obtain *r*, and it is even. (If *r* is odd, see step 5.) Now $b \equiv a^{r/2} \pmod N$ is a square root of 1 modulo N, different from 1. This is because r is the order of a modulo N, so $a^{r/2} \not\equiv 1 \pmod N$, else the order of a in this group would be $r/2$. If $a^{r/2} \equiv -1 \pmod N$, by step 6 we have to restart the algorithm with a different random number a.

Eventually, we must hit an a, of order r in G, such that $b \equiv a^{r/2} \not\equiv 1, -1 \pmod N$. This is because such a b is a square root of 1 modulo N, other than 1 and -1, whose existence is guaranteed by the Chinese remainder theorem, since N is not a prime power.

We claim that $d = \gcd(b-1, N)$ is a proper factor of N, that is, $d \neq 1, N$. In fact if $d = N$, then N divides $b-1$, so that $b \equiv 1 \pmod N$, against the construction of b. If on the other hand $d = \gcd(b-1, N) = 1$, then by Bézout's identity there are integers u, v such that

$$(b-1)u + Nv = 1.$$

Multiplying both sides by $b+1$ we obtain

$$(b^2-1)u + N(b+1)v = b+1.$$

Since N divides $b^2 - 1 \equiv a^r - 1 \pmod N$, we obtain that N divides $b+1$, so that $b \equiv -1 \pmod N$, again contradicting the construction of b.

Thus d is the required proper factor of N.

Finding the period [edit]

Shor's period-finding algorithm relies heavily on the ability of a quantum computer to be in many states simultaneously. Physicists call this behavior a "superposition" of states. To compute the period of a function *f*, we evaluate the function at all points simultaneously.

Quantum physics does not allow us to access all this information directly, though. A measurement will yield only one of all possible values, destroying all others. If not for the no cloning theorem, we could first measure *f*(*x*) without measuring *x*, and then make a few copies of the resulting state (which is a superposition of states all having the same *f*(*x*)). Measuring *x* on these states would provide different *x* values which give the same *f*(*x*), leading to the period. Because we cannot make exact copies of a quantum state, this method does not work. Therefore we have to carefully transform the superposition to another state that will return the correct answer with high probability. This is achieved by the quantum Fourier transform.

Shor thus had to solve three "implementation" problems. All of them had to be implemented "fast", which means that they can be implemented with a number of quantum gates that is polynomial in $\log N$.

1. Create a superposition of states. This can be done by applying Hadamard gates to all qubits in the input register. Another approach would be to use the quantum Fourier transform (see below).
2. Implement the function *f* as a quantum transform. To achieve this, Shor used repeated squaring for his modular exponentiation transformation. It is important to note that this step is more difficult to implement than the quantum Fourier transform, in that it requires ancillary qubits and substantially more gates to accomplish.
3. Perform a quantum Fourier transform. By using controlled rotation gates and Hadamard gates, Shor designed a circuit for the quantum Fourier transform (with $Q = 2^q$) that uses just $q(q-1)/2 = O((\log Q)^2)$ gates.[12]

After all these transformations a measurement will yield an approximation to the period *r*. For simplicity assume that there is a *y* such that *yr*/*Q* is an integer. Then the probability to measure *y* is 1. To see that we notice that then

$$e^{-2\pi i byr/Q} = 1$$

for all integers *b*. Therefore the sum whose square gives us the probability to measure *y* will be *Q*/*r* since *b* takes roughly *Q*/*r* values and thus the probability is $1/r^2$. There are *r* *y* such that *yr*/*Q* is an integer and also *r* possibilities for $f(x_0)$, so the probabilities sum to 1.

Note: another way to explain Shor's algorithm is by noting that it is just the quantum phase estimation algorithm in disguise.

The bottleneck [edit]

The runtime bottleneck of Shor's algorithm is quantum modular exponentiation, which is by far slower than the quantum Fourier transform and classical pre-/post-processing. There are several approaches to constructing and optimizing circuits for modular exponentiation. The simplest and (currently) most practical approach is to mimic conventional arithmetic circuits with reversible gates, starting with ripple-carry adders. Knowing the base and the modulus of exponentiation facilitates further optimizations.[13][14] Reversible circuits typically use on the order of n^3 gates for n qubits. Alternative techniques asymptotically improve gate counts by using quantum Fourier transforms, but are not competitive with less than 600 qubits due to high constants.

Discrete logarithms [edit]

Given prime p with generator g where $1 < g < p - 1$, suppose we know that $x = g^r \pmod p$, for some *r*, and we wish to compute *r*, which is the discrete logarithm: $r = \log_g x \pmod p$. Consider the abelian group $(\mathbb{Z}_p)^\times \times (\mathbb{Z}_p)^\times$ where each factor corresponds to modular multiplication of nonzero values, assuming p is prime. Now, consider the function

$$f(a, b) = g^a x^{-b} \pmod p.$$

This gives us an abelian hidden subgroup problem, as *f* corresponds to a group homomorphism. The kernel corresponds to modular multiples of (*r*,1). So, if we can find the kernel, we can find *r*.

In popular culture [edit]

On the television show *Stargate Universe*, the lead scientist, Dr. Nicholas Rush, hoped to use Shor's algorithm to crack *Destiny*'s master code. He taught a quantum cryptography class at the University of California, Berkeley, in which Shor's algorithm was studied.

Shor's algorithm was also a correct answer to a question in a Physics Bowl competition in the episode "The Bat Jar Conjecture" of the TV series *The Big Bang Theory*.

References [edit]

1. ^ See also Pseudo-polynomial time.
2. ^ http://arxiv.org/abs/quant-ph/9602016 - Efficient Networks for Quantum Factoring
3. ^ MathWorld: Number Field Sieve
4. ^ Vandersypen, Lieven M. K.; Steffen, Matthias; Breyta, Gregory; Yannoni, Costantino S.; Sherwood, Mark H. & Chuang, Isaac L. (2001), "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance" (PDF), *Nature* **414** (6866): 883–887, arXiv:quant-ph/0112176, Bibcode:2001Natur.414..883V, doi:10.1038/414883a, PMID 11780055
5. ^ Lu, Chao-Yang; Browne, Daniel E.; Yang, Tao & Pan, Jian-Wei (2007), "Demonstration of a Compiled Version of Shor's Quantum Factoring Algorithm Using Photonic Qubits", *Physical Review Letters* **99** (25): 250504, arXiv:0705.1684, Bibcode:2007PhRvL..99y0504L, doi:10.1103/PhysRevLett.99.250504
6. ^ Lanyon, B. P.; Weinhold, T. J.; Langford, N. K.; Barbieri, M.; James, D. F. V.; Gilchrist, A. & White, A. G. (2007), "Experimental Demonstration of a Compiled Version of Shor's Algorithm with Quantum Entanglement", *Physical Review Letters* **99** (25): 250505, arXiv:0705.1398, Bibcode:2007PhRvL..99y0505L, doi:10.1103/PhysRevLett.99.250505
7. ^ http://arxiv.org/pdf/1202.5707v1.pdf - Computing prime factors with a Josephson phase qubit quantum processor
8. ^ Martín-López, Enrique; Enrique Martín-López; Anthony Laing; Thomas Lawson; Roberto Alvarez; Xiao-Qi Zhou; Jeremy L. O'Brien (12 October 2012). "Experimental realization of Shor's quantum factoring algorithm using qubit recycling". *Nature Photonics*. arXiv:1111.4147. Bibcode:2012NaPho...6..773M. doi:10.1038/nphoton.2012.259. Retrieved October 23, 2012.
9. ^ Nanyang Xu; Jing Zhu; Dawei Lu; Xianyi Zhou; Xinhua Peng; Jiangfeng Du (30 March 2012). "Quantum Factorization of 143 on a Dipolar-Coupling Nuclear Magnetic Resonance System". *Physical Review Letters* (American Physical Society). doi:10.1103/PhysRevLett.108.130501.
10. ^ Zyga, Lisa (28 November 2014). "New largest number factored on a quantum device is 56,153". *Phys.org* (Science X Network). Retrieved 4 August 2015.
11. ^ A bot will complete this citation soon. Click here to jump the queue arXiv:1411.6758.
12. ^ Shor 1999, p. 14.
13. ^ Markov, Igor L.; Saeedi, Mehdi (2012). "Constant-Optimized Quantum Circuits for Modular Multiplication and Exponentiation". *Quantum Information and Computation* **12** (5–6): 361–394. arXiv:1202.6614. Bibcode:2012arXiv1202.6614M.
14. ^ Markov, Igor L.; Saeedi, Mehdi (2013). "Faster Quantum Number Factoring via Circuit Synthesis". *Phys. Rev. A*

87: 012310. arXiv:1301.3210. Bibcode:2013PhRvA..87a2310M. doi:10.1103/PhysRevA.87.012310.

Further reading [edit]

This article includes a list of references, but **its sources remain unclear** because it has **insufficient inline citations**. Please help to improve this article by introducing more precise citations. *(September 2010)*

- Nielsen, Michael A. & Chuang, Isaac L. (2000), *Quantum Computation and Quantum Information*, Cambridge University Press.
- Phillip Kaye, Raymond Laflamme, Michele Mosca, *An introduction to quantum computing*, Oxford University Press, 2007, ISBN 0-19-857049-X
- "Explanation for the man in the street" by Scott Aaronson, "approved" by Peter Shor. (Shor wrote "Great article, Scott! That's the best job of explaining quantum computing to the man on the street that I've seen."). An alternate metaphor for the QFT was presented in one of the comments. Scott Aaronson suggests the following 12 references as further reading (out of "the $10^{10^{5000}}$ quantum algorithm tutorials that are already on the web."):
- Shor, Peter W. (1997), "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM J. Comput.* **26** (5): 1484–1509, arXiv:quant-ph/9508027v2, Bibcode:1999SIAMR..41..303S, doi:10.1137/S0036144598347011. Revised version of the original paper by Peter Shor ("28 pages, LaTeX. This is an expanded version of a paper that appeared in the Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, Nov. 20--22, 1994. Minor revisions made January, 1996").
- Quantum Computing and Shor's Algorithm, Matthew Hayward's Quantum Algorithms Page, 2005-02-17, imsa.edu, LaTeX2HTML version of the original LaTeX document, also available as PDF or postscript document.
- Quantum Computation and Shor's Factoring Algorithm, Ronald de Wolf, CWI and University of Amsterdam, January 12, 1999, 9 page postscript document.
- Shor's Factoring Algorithm, Notes from Lecture 9 of Berkeley CS 294-2, dated 4 Oct 2004, 7 page postscript document.
- Chapter 6 Quantum Computation, 91 page postscript document, Caltech, Preskill, PH229.
- Quantum computation: a tutorial by Samuel L. Braunstein.
- The Quantum States of Shor's Algorithm, by Neal Young, Last modified: Tue May 21 11:47:38 1996.
- III. Breaking RSA Encryption with a Quantum Computer: Shor's Factoring Algorithm, Lecture notes on Quantum computation, Cornell University, Physics 481-681, CS 483; Spring, 2006 by N. David Mermin. Last revised 2006-03-28, 30 page PDF document.
- arXiv quant-ph/0303175 Shor's Algorithm for Factoring Large Integers. C. Lavor, L.R.U. Manssur, R. Portugal. Submitted on 29 Mar 2003. This work is a tutorial on Shor's factoring algorithm by means of a worked out example. Some basic concepts of Quantum Mechanics and quantum circuits are reviewed. It is intended for non-specialists which have basic knowledge on undergraduate Linear Algebra. 25 pages, 14 figures, introductory review.
- arXiv quant-ph/0010034 Shor's Quantum Factoring Algorithm, Samuel J. Lomonaco, Jr, Submitted October 9, 2000, This paper is a written version of a one hour lecture given on Peter Shor's quantum factoring algorithm. 22 pages.
- Chapter 20 Quantum Computation, from *Computational Complexity: A Modern Approach*, Draft of a book: Dated January 2007, Comments welcome!, Sanjeev Arora and Boaz Barak, Princeton University.
- A Step Toward Quantum Computing: Entangling 10 Billion Particles, from "Discover Magazine", Dated January 19, 2011.
- Josef Gruska - *Quantum Computing Challenges* also in Mathematics unlimited: 2001 and beyond, Editors Björn Engquist, Wilfried Schmid, Springer, 2001, ISBN 978-3-540-66913-5

v · t · e	**Quantum information science**	[show]

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: Quantum algorithms | Integer factorization algorithms | Quantum information science | Post-quantum cryptography