Pollard's rho algorithm

From Wikipedia, the free encyclopedia

Pollard's rho algorithm is a general-purpose integer factorization algorithm. It was invented by John Pollard in 1975.[1] It is particularly effective at splitting composite numbers with small factors.

Contents

Core ideas

The rho algorithm is based on Floyd's cycle-finding algorithm and on the observation that (as in the birthday problem) two numbers x and y are congruent modulo p with probability 0.5 after $1.177\sqrt{p}$ numbers have been randomly chosen. If p is a factor of n, the integer we are aiming to factor, then $p \leq \gcd(x-y, n) \leq n$ since p divides both $x - y$ and n.

The rho algorithm therefore uses a function modulo n as a generator of a pseudo-random sequence. It runs one sequence twice as "fast" as the other; i.e. for every iteration made by one copy of the sequence, the other copy makes two iterations. Let x be the current state of one sequence and y be the current state of the other. The GCD of $|x - y|$ and n is taken at each step. If this GCD ever comes to n, then the algorithm terminates with failure, since this means $x = y$ and therefore, by Floyd's cycle-finding algorithm, the sequence has cycled and continuing any further would only be repeating previous work.

Algorithm

The algorithm takes as its inputs n, the integer to be factored; and f, a function with the property that $x=y \bmod p$ implies $f(x)=f(y) \bmod p$. In the original algorithm, $f(x) = x^2 - 1 \pmod{n}$. The output is either a non-trivial factor of n, or failure. It performs the following steps:[2]

1. $x \leftarrow 2, y \leftarrow 2; d \leftarrow 1$
2. While $d = 1$:
    1. $x \leftarrow f(x)$
    2. $y \leftarrow f(f(y))$

    3. $d \leftarrow \text{GCD}(|x - y|, n)$
3. If $d = n$, return failure.
4. Else, return d.

Note that this algorithm may not find the factors and will return failure for composite n. In that case, use a different $f(x)$ and try again. Note, as well, that this algorithm does not work when n is a prime number, since, in this case, d will be always 1. The algorithm is so-called because the values of f enter a period (mod d), resulting in a ρ shape when diagrammed.

Variants

In 1980, Richard Brent published a faster variant of the rho algorithm. He used the same core ideas as Pollard but a different method of cycle detection, replacing Floyd's cycle-finding algorithm with the related Brent's cycle finding method.[3]

A further improvement was made by Pollard and Brent. They observed that if $\gcd(a, n) > 1$, then also $\gcd(ab, n) > 1$ for any positive integer b. In particular, instead of computing $\gcd(|x - y|, n)$ at every step, it suffices to define z as the product of 100 consecutive $|x - y|$ terms modulo n, and then compute a single $\gcd(z, n)$. A major speed up results as 100 *gcd* steps are replaced with 99 multiplications modulo n and a single *gcd*. Occasionally it may cause the algorithm to fail by introducing a repeated factor, for instance when n is a square. But it then suffices to go back to the previous gcd term, where $\gcd(z, n) = 1$, and use the regular Rho algorithm from there.

Application

The algorithm is very fast for numbers with small factors, but slower in cases where all factors are large. The rho algorithm's most remarkable success has been the factorization of the eighth Fermat number (F_8) by Pollard and Brent. They used Brent's variant of the algorithm, which found a previously unknown prime factor. The complete factorization of F_8 took, in total, 2 hours on a UNIVAC 1100/42.

Example factorization

Let n = 8051 and $f(x) = (x^2 + 1)$ mod 8051.

i	x_i	y_i	GCD($\|x_i - y_i\|$, 8051)
1	5	26	1
2	26	7474	1
3	677	871	97

97 is a non-trivial factor of 8051. Other values of c may give the cofactor (83) instead of 97.

Complexity

The algorithm offers a trade-off between its running time and the probability that it finds a factor. If the squaring function used in the Pollard rho method were replaced by a random function, it would follow that, for all n, running the algorithm for $O(n^{1/4})$ steps would yield a factor with probability at most 1/2. It is believed that the same analysis applies as well to the actual rho algorithm, but this is a heuristic claim, and rigorous analysis of the algorithm remains open.[4]

References

1. ^ Pollard, J. M. (1975), "A Monte Carlo method for factorization", *BIT Numerical Mathematics* **15** (3): 331–334
2. ^ Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. & Stein, Clifford (2001), "Section 31.9: Integer factorization", *Introduction to Algorithms* (Second ed.), Cambridge, MA: MIT Press, pp. 896–901, ISBN 0-262-03293-7 (this section discusses only Pollard's rho algorithm).
3. ^ Brent, Richard P. (1980), "An Improved Monte Carlo Factorization Algorithm" (http://maths-people.anu.edu.au/~brent/pub/pub051.html), *BIT* **20**: 176–184, doi:10.1007/BF01933190 (http://dx.doi.org/10.1007%2FBF01933190)
4. ^ Galbraith, Steven D. (2012), "14.2.5 Towards a rigorous analysis of Pollard rho" (http://books.google.com/books?id=owd76BElvosC&pg=PA272), *Mathematics of Public Key Cryptography*, Cambridge University Press, pp. 272–273, ISBN 9781107013926.

Additional reading

- Katz, Jonathan; Lindell, Yehuda (2007), "Chapter 8", *Introduction to Modern Cryptography*, CRC Press

External links

- Weisstein, Eric W., "Pollard rho Factorization Method (http://mathworld.wolfram.com/PollardRhoFactorizationMethod.html)", *MathWorld*.
- Java Implementation (http://www.cs.princeton.edu/introcs/78crypto/PollardRho.java.html)

Categories: Integer factorization algorithms
