How practical is the Pollard-Strassen Method?

Lillian Zeng, Dr. Ghaith A. Hiary

The Ohio State University – Department of Mathematics

Abstract

The Pollard-Strassen Method is an integer factorization algorithm. It is a deterministic and proven method that terminates in about $n^{\frac{1}{4}+o(1)}$ steps, where n is the integer being factored. This method requires fast multipoint evaluation and high-precision fast Fourier transform for integer multiplication. However, its critics argue that it is less efficient in practice and requires large memory space compared to other integer factorization algorithms. We implemented the Pollard-Strassen Method using C++ programming and GNU MP Library, and tested its running time. It turns out to be the most efficient algorithm among all deterministic algorithms for integer factorization that we tested once n has 30 digits or more.

Objectives

It is of interest to find the most efficient deterministic algorithm for integer factorization in different scenarios. This project provides a full implementation of Pollard Strassen Method, which is the fastest deterministic algorithms for integer factorization, and tested against others.

Pollard Strassen Method for Integer Factorization

- **The algorithm**

Input: an positive integer n
Output: a nontrivial factor of n, or return n is prime
Steps:

1. $b \leftarrow \lfloor\sqrt{n}\rfloor$; $c \leftarrow \lfloor\sqrt{b}\rfloor$; $k \leftarrow \log_2 c$
2. Compute the coefficients of
$$f(x) = \prod_{1\le j\le c}(x+j) = (x+1)(x+2)(x+3)\dots(x+c)$$
3. Call fast multipoint evaluation algorithm to compute $g_i = f(ic) \bmod n$ for $0 \le i < c$
4. Compute GCD (g_i, n) for each g_i until GCD$(g_i,n) > 1$ for some g_i, then GCD(g_i,n) is a factor of n;
If GCD$(g_i,n) = 1$ for all g_i, then return n is prime

- **Main Idea**

From the above algorithm, we know that $g_i = f(ic) = \frac{(ic+c)!}{(ic)!} \bmod n$ for all integer $i \in [0, c)$;
By calculating the GCD(g_i,n) for each g_i the algorithm search through each block: $[0, c]$, $[c+1, 2c]$, $[2c+1, 3c]$…$[(c-2)c+1, (c-1)c]$

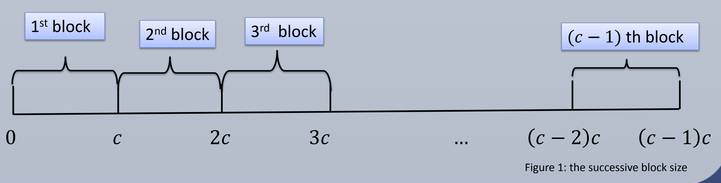

Figure 1: the successive block size

Main Techniques

- **Fast Multipoint Evaluation**

In step 3, we used fast multipoint evaluation to evaluate a polynomial $f(x) = \prod_{1\le j\le c}(x+j) = (x+1)(x+2)(x+3)\dots(x+c)$ with degree of c. It can be performed in approximately $O(M(c)\log c)$ operations.

For a polynomial $h(x) = \prod_{0\le j<\ell}(x+u_j) = (x+u_0)(x+u_1)(x+u_2)\dots(x+u_{\ell-1})$, where $\ell = 2^k$, the algorithm splits the point set $\{u_0, u_1, \dots, u_{\ell-1}\}$ into two halves of equal cardinality and to proceed recursively with each of the two halves. See below for the binary tree demonstration of depth $\log_2 \ell$ with tree root $\{u_0, u_1, \dots, u_{\ell-1}\}$ and the leaves $u_0, \dots, u_{\ell-1}$.

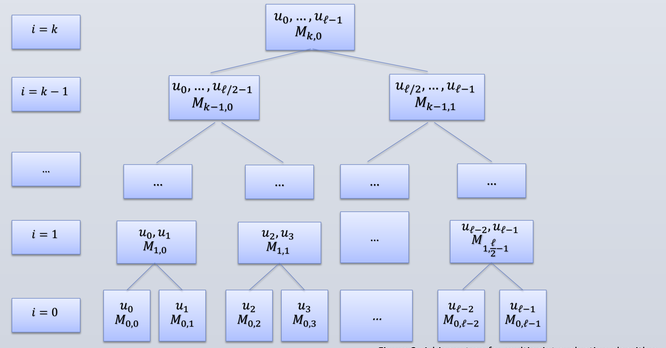

Figure 2: A binary tree for multipoint evaluation algorithm

- **Fast Fourier Transform**

Fast Fourier Transform (FFT) is another crucial technique for the Pollard Strassen Method to reach a complexity of $n^{\frac{1}{4}+o(1)}$ steps, where n is the integer being factored. It was used to multiply two polynomials in step 2. Using FFT, it only takes $\Theta(\ell \lg \ell)$ steps to multiply two polynomials $A(x)$ and $B(x)$ of degree $\ell - 1$, but a naïve way would take $\Theta(\ell^2)$. Below is a comparison between two algorithms for polynomial multiplication.

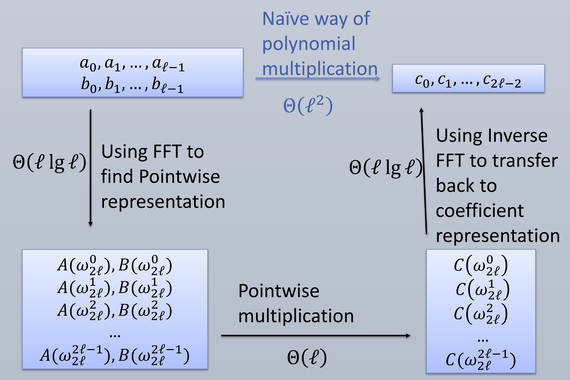

Figure 3: a comparison between two algorithms for polynomial multiplication

Result: Comparison between Integer Factorization Algorithms

Pollard Strassen method is the fastest deterministic algorithm for integer factorization, but it requires heavy memory space. To factor a 17-digit n, the algorithm needs 600MB memory space, and to factor a 19-digit n, it needs about 2.5GB. Pollard Rho method is also a fast algorithm, but its running is probabilistic only, because the algorithm involves probabilistic choices that are hard to predict completely.

Algorithm	Running Time	Memeory Space
Trial Division Method	$O(n^{\frac{1}{2}})$	$O(1)$
Fermat Method	between $O(\log n)$ and $O(n)$ depending on the distance between prime factors	$O(1)$
Pollard Rho Method	$O(n^{\frac{1}{4}})$ (heuristic)	$O(1)$
Pollard Strassen Method	$n^{\frac{1}{4}+o(1)}$	$n^{\frac{1}{4}+o(1)}$

Table 1: a comparison between integer factorization algorithms

Below is a speed competition for integer factorization. Numbers from 15 digits to 23 digits were tested. The Pollard Strassen method is much more efficient than trial division and Fermat method with large integer n. In practice, Pollard Rho method runs even faster, but it is probabilistic.

Digits of n	Trial Division Method	Fermat Method	Pollard Strassen Method	$k = \log_2(n^{0.25})$ in Pollard Strassen Method
15	0.05s	0.15s	28.87s	13
17	0.46s	1.38s	1m19.26s	14
19	4.25s	11.86s	12m58.34s*	16
21	47.06s	2m35.87s	13m0.06s	16
23	7m53.46s	22m58.03s	42m(extrapolation)	-
25	1h20m(extrapolation)	3h40m(extrapolation)	2h15m(extrapolation)	-
27	13h20m(extrapolation)	1d13h(extrapolation)	7h(extrapolation)	-

Table 2: speed comparison between integer factorization algorithms

*The 19-digit n with $k = 16$ resulted in a running time of 12m58.34s, which is about 10 times longer than that of the 17-digit n tested. Because the running time for Pollard Strassen method is based on k, a test for a 19-digit n with $k = 15$ took 3m57.03s to operate, which is only about 3 times longer.

References

[1] Cormen, Thomas H. *Introduction to Algorithms*. Cambridge, Mass: MIT Press, 2009. Internet resource.

[2] Crandall, Richard E, and Carl Pomerance. *Prime Numbers: A Computational Perspective.* New York, NY: Springer, 2005. Internet resource.

[3] Hiary, Ghaith A. *A Deterministic Algorithm for Integer Factorization.* Mathematics of Computation. 85.300 (2016). Print.

[4] Gathen, Joachim. *Modern Computer Algebra*. Cambridge University Press, 2013. Web. 24 Mar. 2017.