Elementary Algorithms in Number Theory

Lillian Zeng, Dr. Ghaith A. Hiary

The Ohio State University – Department of Mathematics

Abstract

This research project summarized and implemented four representative integer factorization algorithms, including Fermat Method, Trial Division, Pollard Rho Method and Pollard–Strassen Method. Their basic ideas and characteristics were introduced, respectively. To test the different features of these algorithms, we implemented them using C++ programming and GNU MP Library. The comparison of these algorithms shows that, if $n = p1 * p2$, then the worst cases in Fermat Method are the best ones in Trial Division, and vice versa, depending on the distance between the two prime factors $p1$ and $p2$; Pollard Rho Method provides an efficient algorithm, which takes only about one second to factor a number with 25 digits, where Fermat Method and Trial Division would take several hours. However, the running time of the Pollard Rho Method has not been proven, so we tested another algorithm, the Pollard–Strassen Method, which is a deterministic and proven method that has the same theoretical running time as the Pollard Rho Method, but is less efficient in practice and requires large memory space. This project hopes to provide a reference for future works to find a more efficient deterministic technique for integer factorization.

Objectives

As several previous works have considered finding a more efficient deterministic technique for integer factorization, it is of interest to provide a comparison among the already known algorithms. This project hopes to provide such a comparison for future works.

Trial Division & Fermat Method

- Trial Division

Trial Division is a basic algorithm for integer factorization. For an integer n, it sequentially tests each integer in $[2, \sqrt{n}\,]$ to see if it is divisible by n. It takes $O(\sqrt{n})$ steps to factor an integer n.

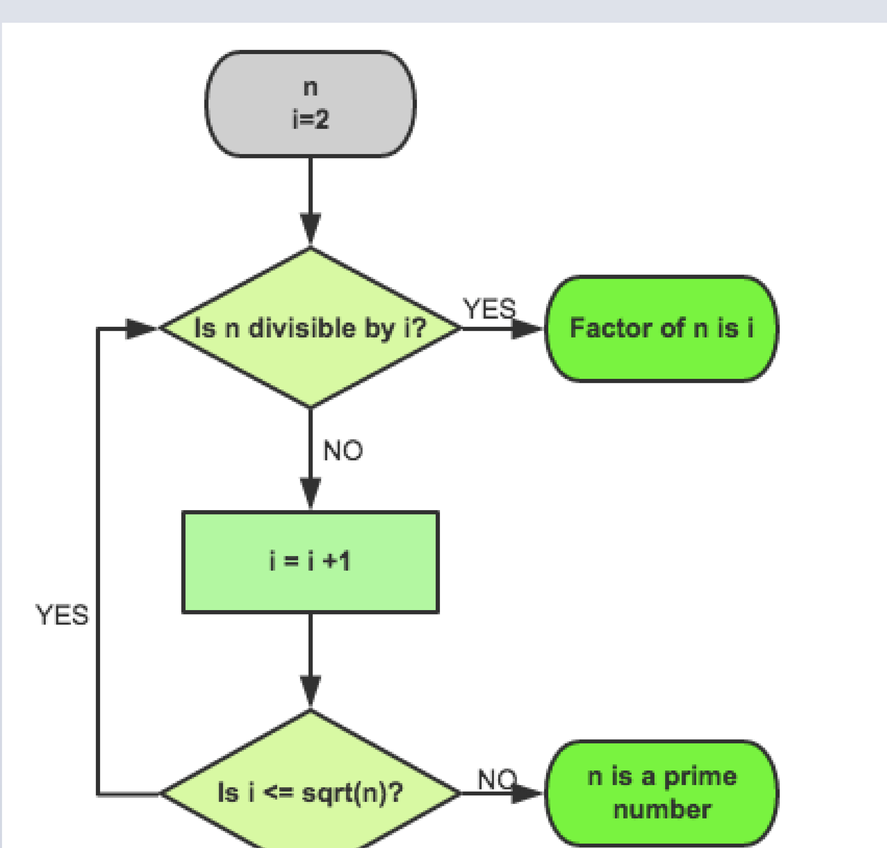

Figure 1. The algorithm of trial division

- Fermat Method

Fermat Method for integer factorization is another basic algorithm. It rewrites the odd integer $n = p1 * p2$ as $n = a^2 - b^2 = (a + b)(a - b)$, where a and b are both integers and $a = \frac{p2+p1}{2}$, $b = \frac{|p2-p1|}{2}$ for factors $p1$ and $p2$. Fermat Method takes $O(p1 - p2)$ steps to complete. So if $p1$ and $p2$ are close to each other, then Fermat method finishes quickly.

- Different choices of distance between the two factors

The greater the distance between the two factors, the longer it will take to find them using Fermat Method. Also, the easiest cases in the Fermat Method are the worst cases in Trial Division, and vice versa.

Tests on different choices of distances between two factors

n (digits)	p_1 (digits)	p_2 (digits)	$p_2 - p_1$	Time
951903871 (9)	27449 (5)	34679 (5)	7230	0.000s
16487873177 (11)	27449 (5)	600673 (6)	573224	0.008s
54040246199 (11)	27449 (5)	1968751 (7)	1941302	0.024s
2068889678717 (13)	27449 (5)	75372133 (8)	75344684	1.128s
18303231155381 (14)	27449 (5)	666808669 (9)	666781220	10.256s
129340249908479 (15)	27449 (5)	4712020471 (10)	4711993022	1min19.400s
308610717679871 (15)	27449 (5)	11243058679 (11)	11243031230	3min14.160s
15249383415859739 (17)	27449 (5)	555553332211 (12)	555553304762	2hr46min47.504s

Table 1. Speed test for Fermat Method

Below is a graph for the speed tests of Fermat Method. $\log\left(\frac{p2-p1}{(p2-p1)_0}\right)$ were calculated for x-axis, and $\frac{\log\left(\frac{time}{time_0}\right)}{\log\left(\frac{p2-p1}{(p2-p1)_0}\right)}$ were calculated for y-axis. The graph looks like a horizontal line around 1, meaning that the running time increases linearly with $p1 - p2$.

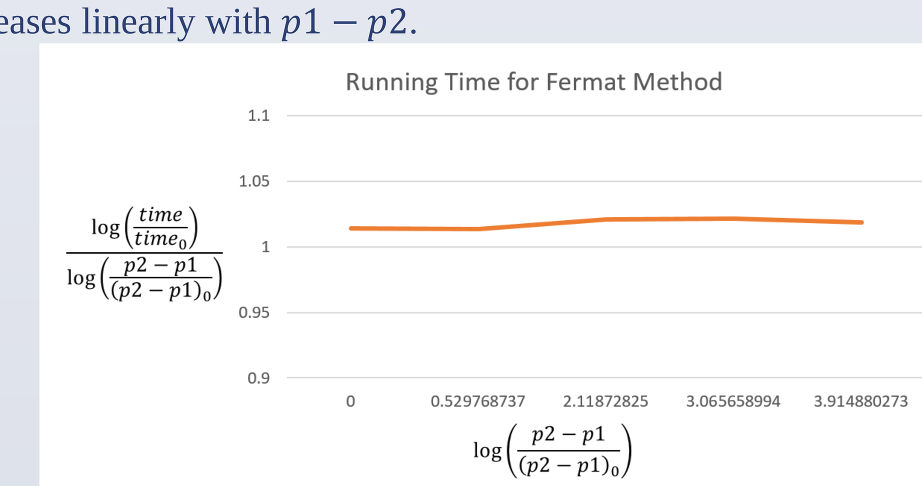

Figure 2. Speed tests for Fermat Method

Pollard Rho Method

Pollard Rho Method is an algorithm for integer factorization using the idea of birthday paradox. It terminates in $O(\sqrt{p})$ steps, where p is the least prime factor of n.

- The Birthday Paradox

In a room with 23 people, there is about 50 percent chance that two of them have the same birthday.

$$
\begin{aligned}
&probability\ (two\ of\ them\ have\ the\ same\ birthday) \\
&= 1 - probability\ (none\ of\ them\ have\ the\ same\ birthday) \\
&= 1 - \frac{365}{365} * \frac{364}{365} * \frac{363}{365} * \cdots * \frac{365-23+1}{365} \\
&= 1 - \frac{365!}{365^{23} * (365-23)!} \\
&\approx\ 0.5073
\end{aligned}
$$

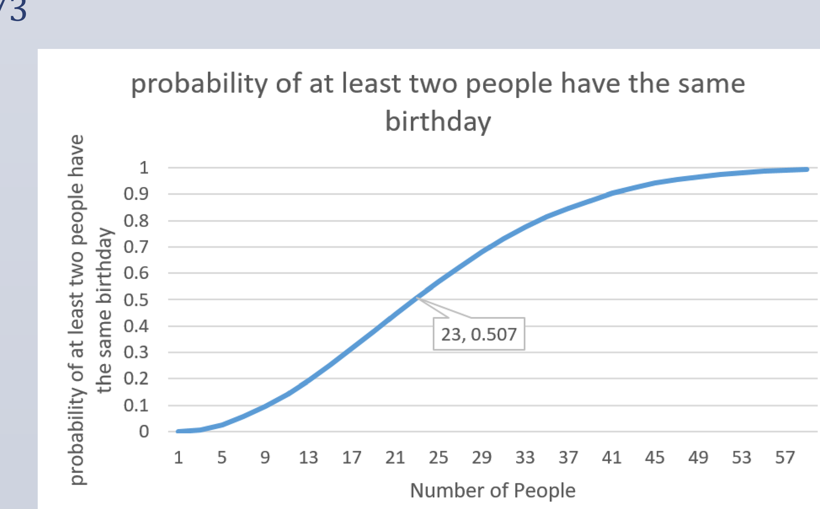

Figure 3. Probability of at least two people have the same birthday with different number of people in the room

Applying Birthday paradox in Pollard Rho Method (example)

Let $p = 29$

Let x be in interval [1,1000]

We want to know the probability of x = p using different methods to get x

Method 1: let x = a random number in [1,1000]

probability of $x = p$: $\frac{1}{1000}$

Method 2: choosing two random numbers in [1,1000]

let x = the difference of the two numbers

probability of $x = p$: $\frac{2*(1000-29)}{1000*1000} \approx 0.0019 \approx \frac{1}{500}$

Method 3: Birthday paradox:

choosing a set of random numbers (more than 2 numbers) in [1,1000]

let x = the difference of any two number in the set of numbers

the probability of $x = p$ will increase very quickly

- Different Choices of $F(x)$

In implementations, the choice of function $F(x)$ matters, where $F(x)$ is a random map. Below is a table that shows some tests made on different choices of $F(x)$. Some of the bad choices include $ax + b$, because it is not random enough that a linear function is very predictable .

Tests on different choices of $F(x)$ (record the number of iterations)

n	p	x^2+a	x^2+ax+b	x^3+a	x^4+a	x^2	$ax+b$	x^2-2
304,583	541	11	15	14	16	12	180	36
56,675,701	541	27	18	10	14	18	135	4
8,377,886,507	541	23	18	12	24	12	180	12
1,102,598,450,029	541	20	18	4	8	36	108	135
10,971,514,769	104,729	112	149	230	182	48	52,380	144
239,812,943,950,001	15,485,863	3,334	5,528	6,716	2,330	30,900	7,742,963	8,838
4,153,748,711,044,459,367	2,038,074,743	9,158	19,583	8,428	7,040	90,090	679,358,256	771,210
63,553,301,090,046,162,415,541	252,097,800,667	510,260	278,203	126,255	198,210			

Table 2. Speed tests for different choices of $F(x)$

- Speed Test of Pollard Rho Method

Tests on Pollard Rho Method

n (digits)	p_1 (digits)	p_2 (digits)	# of iterations	Time
2422167482706164250688963 (25)	1398023584459 (13)	1732565537257 (13)	850612	0.368s
481091880198991122823285789 (27)	11108452651921 (14)	43308631298509 (14)	2859233	1.372s
90160752738524403660909417347 (29)	277777788888989 (15)	324578696875423 (15)	22689861	11.376s
6098538498637700801384373932681 (31)	2222226095589337 (16)	2744337540964913 (16)	44755082	24.340s
486597680313197081893904767621601 (33)	13107181911273173 (17)	37124508045065437 (17)	67631786	37.844s
63703751283340493398721535334589851 (35)	212345678987654321 (18)	300000224101777931 (18)	374537991	3min37.568s
6763125248573334830173265909231499061 (37)	1111918171614121013 (19)	6082394749206781697 (19)	786228959	8min9.032s
625030198653794594481152214235491584911 (39)	17131175322357111317 (20)	36484957213536676883 (20)	7639357254	1hr53min55.448s
79754154553006699908253549560504465309943 (41)	223331355555777767777 (21)	357111317192931414359 (21)	4109561190	1hr19m52.108s
4110649089119698872902667000974290101778663 (43)	2030507011013017019023 (22)	2024444666688888688681 (22)	56854345256	19hr22m41.812s
415729465994717803184678486449392367860085511 (45)	16081123194129907217449 (23)	25852016738884976645039 (23)	153343425024	2d5hr55m51.816s

Table 3. Speed tests for the Pollard Rho Method

Below is a graph illustration of the speed tests of Pollard Rho Method. We use $\log\left(\frac{p}{p_0}\right)$ as the measurement for x-axis, and $\frac{\log\left(\frac{time}{time_0}\right)}{\log\left(\frac{p}{p_0}\right)}$ as the measurement for y-axis. As we can see, the graph is a constant line slightly lower than 0.25.

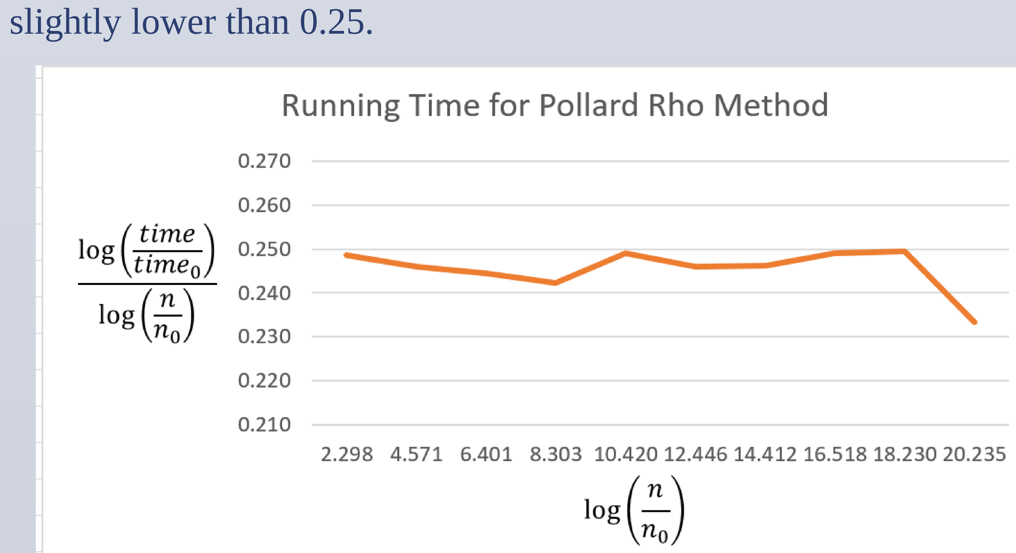

Figure 4. Speed tests for the Pollard Rho Method

- Application of Pollard Rho Method

An application of Pollard Rho Method is to factor Fermat numbers. For the number $F5 = 2^{2^5} + 1$ (i.e. 4,294,967,297), it only iterates once to get factor; and for the number $F6 = 2^{2^6} + 1$ (i.e. 8,446,744,073,709,551,617), it iterates 3 times to get factor.

- Features of Pollard Rho Method

Pollard Rho Method requires very little space (i.e. $O(1)$) and a small amount of operations (i.e. $O(n^{\frac{1}{4}})$). However, the running time of the Pollard Rho Method has not been fully proven, because the algorithm involves probabilistic choices that are hard to predict completely.

Pollard Strassen Method

The Pollard-Strassen Method is the fastest deterministic algorithm for integer factorization. It terminates in $O(n^{\frac{1}{4}} \ln^2 n)$ steps, However, it requires a memory space of $O(n^{\frac{1}{4}})$. It also looks like a constant line around 0.25 in the graph below.

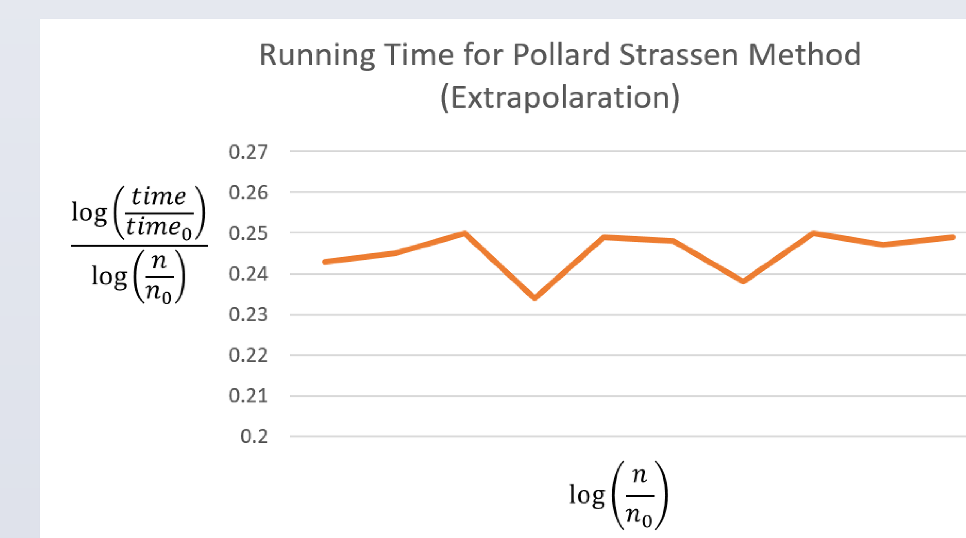

Figure5. Speed tests for the Pollard Rho Method

Result: Speed Comparison between Algorithms

Below is a speed competition for integer factorization algorithms (except Pollard Strassen Method). Numbers from 15 digits to 27 digits were tested. The Pollard Rho Method is much more efficient than Trial Division and Fermat Method.

Running time for different integer factorizing algorithms

n (digits)	p (digits)	Trial Division	Fermat Method	Pollard Rho Method
360576736643657 (15)	10243657 (8)	0.183s	0.172s	0.003s
50881147147252369 (17)	131333131 (9)	2.385s	1.838s	0.004s
3744920489192229499 (19)	1000273817 (10)	18.179s	21.909s	0.040s
353425640213504447887 (21)	10127896543 (11)	3m35.135s	3m5.977s	0.106s
33434808022464851349547 (23)	101740496633 (12)	36m50.061s	27m56.715s	0.200s;
5055578471846736261030703 (25)	1431535622177 (13)	5hr(extrapolation)	5hr(extrapolation)	1.271s
526891015918100301574668559 (27)	15261281789861 (14)	2d2hr(extrapolation)	2d2hr(extrapolation)	1.444s

Table 4. Speed competition for different integer factorization algorithms

References

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

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

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