N.P.F.A.
On a New Variant of Prime Factorization Algorithm

Zhi Ping Ooi, Jin Xian Tee

January 31, 2023

Abstract

For centuries, prime factorization has been a fundamental theoretical problem in the mathematical world. Conventionally, the algorithms for prime number factorization demand a computational cost that grows exponentially with the magnitude of number to be factorized. Existing algorithms will hit a formidable barrier due to the finite computation resource practically available. In this research, we have attempted to introduce a new and effective approach: an improvement of Fermat's Factorization algorithm. Our method mainly involves two variables k and s, $s = \lceil \sqrt{n * k} \rceil$, where k is a natural number. This ensures that in most of the cases, we get a small value from $s^2 \mod n$, which aids in finding factors as the distribution of perfect squares are denser at smaller numbers. Results have shown that our approach is capable of prime factorizing numbers hundreds of times faster than the original method, Fermat's Factorization algorithm, while being almost on par with Pollard's Rho. For all of the numbers aside of certain semiprimes, our algorithm can factorize it under < 0.01 second. Besides, our new method allows us to implement parallel processing while searching for factors, greatly increasing its efficiency. Therefore, this approach is a new gateway to creating an efficient prime factorizing algorithm, hence bringing benefits to the mathematical field.

1 Introduction

Prime numbers are numbers that cannot be decomposed into smaller prime constituents, which also means that the only two divisors for prime numbers are 1 and itself. Factoring a number into its prime factors, also known as prime factorizing is a crucial part of number theory.

The earliest invention in this field are lookup tables published in individual volumes, each with tables of primes factors for numbers within a certain range. However, there is a limit to how many numbers they can publish as tables. Therefore other techniques are required to solve this problem.

The early factorizing algorithms include Trial Division, Fermat's Factorization Algorithm and Euler's Algorithm.

2 Fermat's Factorization Algorithm (FF)

Based on The Fundamental Theorem of Arithmetic, every integer has a unique prime decomposition. Therefore, every odd integer n can be written as pq, $p > q$, where p and q is n and 1 when n is prime. FF was discovered in a letter addressed to Mersenne by Pierre de Fermat in 1643. FF is based on the fact that every odd integer can be represented as a difference of two squares, $n = x^2 - y^2$. Since n is odd, p, q is also odd.

Lenma 2.1. *Let $p = x + y$ and $q = x - y$.*

$$\begin{aligned} p + q &= 2x \\ p - q &= 2y \\ x = \frac{p+q}{2},\ y &= \frac{p-q}{2} \end{aligned}$$

$x, y \in \mathbb{N}$, since the sum and difference of two odd numbers p, q are even.

Proof.

$$\begin{aligned} n &= x^2 - y^2 \\ &= \left(\left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2\right) \\ &= \frac{1}{4}\left((p+q)^2 - (p-q)^2\right) \\ &= \frac{1}{4}(4pq) \\ &= pq \end{aligned}$$

□

FF starts the search of factors from $x = \lfloor \sqrt{n} \rfloor + 1$, and increments x by 1 every iteration. This is based on the fact that $x^2 > n \implies x > \sqrt{n}$.

One of the fastest improvement of FF is Fermat's Sieving with Odd and Even (FOE). This improvement utilizes the property of x being odd and y being even when $n = 4k + 1$ and vice versa when $n = 4k - 1$. This improvement increases the speed of FF to about twice of its original as the value of x now increases by 2 every iteration instead of 1.

3 Theoretical Framework

Let $x, y \in \mathbb{N}, x, y < n$. Instead of using the original concept of FF, where $x^2 - y^2 = n$, a congruence is squares, where $x^2 \equiv y^2 \mod n$ is used. By finding a congruence of squares where $x + y \neq n$, factors of n can still be determined by using the concept of FF.

Proof. If $x + y = n$, $(x - y)$ does not have factor of n if $\gcd(z, n) = 1$.

$$\begin{aligned} x^2 &\equiv y^2 \ (mod\ n) \\ x^2 - y^2 &= nz,\ z \in \mathbb{N} \\ (x+y)(x-y) &= nz \\ (n)(x-y) &= nz \\ x - y &= z \end{aligned}$$

$\therefore \gcd(x - y, n) = 1.$ □

Proof. If $x + y \neq n$, $\gcd(x + y, n)$ and $\gcd(x - y, n) =$ factor of n.

$$(x + y)(x - y) = pqz$$

Based on The Fundamental Theorem of Arithmetic, the factors of n must exist in $(x + y)(x - y)$, so if $p|(x \pm y)$, then $q|(x \mp y)$. □

Instead of using incrementation of x by 1, the search for x is changed to using $\lceil\sqrt{n \cdot k}\rceil$, where $k \in \mathbb{N}$. This method is based on the fact that the amount of perfect squares are denser in smaller values. With this method, the possible y can be generated in a more efficient way. Let $x_i = \lceil\sqrt{n \cdot k}\rceil$, $y_i^2 \equiv x_i^2 \mod n$, $y_i^2 \leq 2x_i$.

Lenma 3.1. *For every* x, $(x)^2 - (x - 1)^2 = x^2 + 2x - 1 - x^2 = 2x - 1$.

Proof. $(\lceil\sqrt{n \cdot k}\rceil)^2$ finds the first perfect square larger than $n \cdot k$, while $(\lfloor\sqrt{n \cdot k}\rfloor)^2$ finds the first perfect square smaller than $n \cdot k$. Therefore, the largest value for $(x_i - 1)^2 \mod n = n - 1$.

$$\begin{aligned} x_i^2 - (x_i - 1)^2 &= 2x_i - 1 \\ x_i^2 &= n - 1 + 2x_i - 1 \\ x_i^2 &= 2x_i - 2 \ (mod\ n) \\ y_i^2 &= 2x_i - 2 \end{aligned}$$

$\therefore$ If $(x_i - 1)^2 \mod n \leq n - 1$, then $y_i^2 \leq 2x_i - 2$. □

Since $x + y = n$, $y = n - x$. Therefore, if $x, y < \frac{1}{2}n$, $x + y < n$, k is set to a maximum value of less than $\lceil \frac{1}{4}n \rceil$.

Proof. If $k < \lceil \frac{1}{4}n \rceil$, $x < \lceil \frac{1}{2}n \rceil$, for $k, x \in \mathbb{N}$.

$$x = \left\lceil \sqrt{n \cdot \frac{1}{4}n} \right\rceil$$
$$x = \left\lceil \sqrt{\frac{n^2}{2^2}} \right\rceil$$
$$x = \left\lceil \frac{1}{2}n \right\rceil$$

$\therefore$ If $k < \lceil \frac{1}{4}n \rceil$, then $x + y < n$. $\square$

4 Methodology

Algorithm 1 $k^2 - x^2$

Require: $n \in \mathbb{N},\ n \geq 2$

$k := 1,\ x := 1$
for $k := 1, k \leq \left\lceil \frac{n}{4} \right\rceil$ **do**
    for $x := 1, k \leq 1000$ **and** $x < k$ **do**
        $k_i := k^2 - x^2$
        $s := \left\lceil \sqrt{n * k_i} \right\rceil$
        $s_{gcd} := \gcd(s, n)$
        if $n > s_{gcd} > 1$ **then**
            print s_{gcd}
        $s'_b := s^2 \bmod n$
        $s'_a := \sqrt{s'_b}$
        if $s'_a \in \mathbb{N}$ **then**
            print $\gcd(|(s - s'_a)|, n)$
            print $\gcd((s + s'_a), n)$
        $x + 1$
    $k + 1$

Algorithm 2 $k+2$

Require: $n \in \mathbb{N},\ n \geq 2$

1: $k := 1$
2: **for** $k := 1, k \leq \left\lceil \frac{n}{4} \right\rceil$ **do**
3: $\quad s := \left\lceil \sqrt{n * k} \right\rceil$
4: $\quad s_{gcd} := \gcd(s, n)$
5: $\quad$ **if** $n > s_{gcd} > 1$ **then**
6: $\quad\quad$ **print** s_{gcd}
7: $\quad s'_b := s^2 \bmod n$
8: $\quad s'_a := \sqrt{s'_b}$
9: $\quad$ **if** $s'_a \in \mathbb{N}$ **then**
10: $\quad\quad$ **print** $\gcd(|(s - s'_a)|, n)$
11: $\quad\quad$ **print** $\gcd((s + s'_a), n)$
12: $\quad k + 2$

Algorithm 3 $k+8$

Require: $n \in \mathbb{N},\ n \geq 2$

1: $k := 1$
2: **for** $k := 8, k \leq \left\lceil \frac{n}{4} \right\rceil$ **do**
3: $\quad s := \left\lceil \sqrt{n * k} \right\rceil$
4: $\quad s_{gcd} := \gcd(s, n)$
5: $\quad$ **if** $n > s_{gcd} > 1$ **then**
6: $\quad\quad$ **print** s_{gcd}
7: $\quad s'_b := s^2 \bmod n$
8: $\quad s'_a := \sqrt{s'_b}$
9: $\quad$ **if** $s'_a \in \mathbb{N}$ **then**
10: $\quad\quad$ **print** $\gcd(|(s - s'_a)|, n)$
11: $\quad\quad$ **print** $\gcd((s + s'_a), n)$
12: $\quad k + 8$

Algorithm 4 $k + k_i$

Require: $n \in \mathbb{N},\ n \geq 2$

$t := 1000, t_i := 100, k_i := 0$
for $k := 1000, k \leq \left\lceil \frac{n}{4} \right\rceil$ **do**
  if $k \geq t$ **then**
    $t_i + 1000, t + t_i$
    $k_i + 10$
  $s := \left\lceil \sqrt{n * k} \right\rceil$
  $s_{gcd} := \gcd(s, n)$
  if $n > s_{gcd} > 1$ **then**
    print s_{gcd}
  $s'_b := s^2 \bmod n$
  $s'_a := \sqrt{s'_b}$
  if $s'_a \in \mathbb{N}$ **then**
    print $\gcd(|(s - s'_a)|, n)$
    print $\gcd((s + s'_a), n)$
  $k + k_i$

5 Future works

For future works, we propose:

- Proving the occurance of a pattern in valid k values
- Proving the reason in which the first k value is always an odd or a multiple of 8

6 Conclusion

In this research, we've introduced a new variant of prime factorization algorithm that outperforms Fermat's Factorization algorithm and it's improvements for numbers smaller than 10^{32}. Through quantitative analysis, we believe that there are more promising patterns waiting to be discovered, and potentially proven.

References

[1] Ward, K. (n.d.). Fermat Method Sieve. https://trans4mind.com/personal_development/mathematics/numberTheory/fermatSieve.htm

[2] Somsuk, K. (2020). The new integer factorization algorithm based on Fermat's Factorization Algorithm and Euler's theorem. International Journal of Electrical and Computer Engineering, 10(2), 1469. https://doi.org/10.11591/ijece.v10i2.pp1469-1476

[3] Huang, Q., Li, Z., Zhang, Y., & Lu, C. (2007). A Modified Non-Sieving Quadratic Sieve For Factoring Simple Blur Integers. In Multimedia and Ubiquitous Engineering. https://doi.org/10.1109/mue.2007.28

[4] Bahig, H. M., Mohammed, A., Khaled, A., AlGhadhban, A., & Hatem, M. (2020). Performance Analysis of Fermat Factorization Algorithms. International Journal of Advanced Computer Science and Applications, 11(12). https://doi.org/10.14569/ijacsa.2020.0111242

[5] Aminudin, A., & Cahyono, E. F. (2021b). A Practical Analysis of the Fermat Factorization and Pollard Rho Method for Factoring Integers. Lontar Komputer, 12(1), 33. https://doi.org/10.24843/lkjiti.2021.v12.i01.p04

[6] Wolfram Research, Inc. (n.d.). Fermat's Factorization Method – from Wolfram MathWorld. https://mathworld.wolfram.com/FermatsFactorizationMethod.html Sharma, P. K., Gupta, A., & Vijay, A. (2012). Notice of Violation of IEEE Publication Principles: Modified Integer Factorization Algorithm Using V-Factor Method. In International Conference on Advanced Computing. https://doi.org/10.1109/acct.2012.73

[7] Ağargün, A., & Ozkan, E. M. (2001). A Historical Survey of the Fundamental Theorem of Arithmetic. Historia Mathematica, 28(3), 207–214. https://doi.org/10.1006/hmat.2001.2318

[8] Smith, F.J., 2006. A brief history of factorization techniques. University of Washington.