Fast Approach to Factorize Odd Integers with Special Divisors

1,2Xingbo Wang and 1Junjian Zhong

1Department of Mechatronic Engineering, Foshan University, Foshan City, PRC, 528000, China
2State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
3Guangdong Engineering Center of Information Security for Intelligent Manufacturing System, China

Abstract: The paper proves that an odd composite integer \( N \) can be factorized in \( O((\log_2 N)^3) \) bit operations if \( N = pq \), the divisor \( q \) is of the form \( 2^\alpha u + 1 \) or \( 2^\alpha u - 1 \) with \( u \) being an odd integer and \( \alpha \) being a positive integer and the other divisor \( p \) satisfies \( 1 < p \leq 2^\alpha + 1 \) or \( 2^\alpha + 1 < p \leq 2^{\alpha+1} - 1 \). Theorems and corollaries are proved with detail mathematical reasoning. Algorithm to factorize the odd composite integers is designed and tested in Maple. The results in the paper demonstrate that fast factorization of odd integers is possible with the help of valuated binary tree.

Keywords: Cryptography, Integer Factorization, Binary Tree, Algorithm

Introduction

A Valuated Binary tree is a full perfect binary tree that has odd integers bigger than 1 put on it from top to bottom and left to right, as introduced in Wang’s (2016a). With the help of the valuated binary tree, many new properties of the odd integers are discovered. For example, the properties of symmetric nodes and symmetric common divisors, the properties of subtree duplication and subtree transition and the properties of sum by level, root division and uniform sum were discovered in (Wang, 2016b; 2017a), the genetic properties of odd integers was disclosed in (Wang, 2017b) and the periodical divisibility traits along the leftmost path or the left side-path of the tree were demonstrated in (Wang and Guo, 2019). All these new properties enable us to know the integers in a different point of view, as stated and investigated in Wang’s (2018). Integer factorization has been a hard problem in number theory and in cryptography over years, as overviewed in Yan’s (2013), Sarnaik’s et al. (2016) and Phulachand’s (2016). Any new approach related with the integers shall of course be tried on the issue. Wang (2017b) proved that there should exist an algorithm of \( O((\log_2 N)^2) \) searching steps to factorize an odd integer \( N \). But there has not been a convincible demonstration. Thereby, this paper, continues the studies on integer factorization and proves that there are odd integers that can be factorized in \( O((\log_2 N)^3) \) searching steps or in \( O((\log_2 N)^4) \) bit operations.

Preliminaries

Definitions and Notations

A valuated binary tree \( T \) is such a binary tree that each of its nodes is assigned a value. An odd number \( N \)-rooted tree, denoted by \( T_N \) is a recursively constructed valuated binary tree whose root is the odd number \( N \) with \( 2N + 1 \) and \( 2N + 1 \) being the root’s left and right sons, respectively. Each son is connected with its father via a path, but there is no path between the two sons. \( T_3 \) tree is the case \( N = 3 \). For convenience, symbol \( N_{(i,j)} \) is by default the node at position \( j \) on level \( k \) of \( T_N \), where \( k = 1, 2, \ldots \) and \( j = 0, 1, \ldots, 2^k - 1 \). Symbol \( N^N_{(i,j)} \) is to denote the node at position \( j \) on level \( k \) of \( T_N \), where \( k = 1, 2, \ldots \) and \( j = 0, 1, \ldots, 2^k - 1 \). Symbol \( X \in l(T_N) \) means node \( X \) is in the left branch of \( T_N \) while symbol \( X \in r(T_N) \) means \( X \) is in the right branch of \( T_N \). Symbol \( N^N_{(i,j)} \) is the leftmost node on level \( i \) of \( T_N \); use symbol \( N^N_{(i-1,j)} \) to denote the odd number left to \( N^N_{(i,j)} \), namely, \( N^N_{(i-1,j)} = N^N_{(i,j)} - 2 \). Use symbol \( P^N_{(i,j)} \) to indicate the leftmost path defined by \( P^N_{(i,j)} = \{ N^N_{(0,0)}, N^N_{(1,0)}, \ldots, N^N_{(i-1,j)} \} \) and symbol \( P^N_{(i,j)} \) to indicate the path defined by \( P^N_{(i,j)} = \{ N_{(1,1)}, \ldots, N_{(i-1,j)}, \ldots \} \), which is also called a left side-path, as depicted in Fig. 1. The leftmost path and the rightmost path together with their side-paths respectively are in all called border-path or simply border.
Symbol \( A \Rightarrow B \) means result \( B \) is derived from condition \( A \) or \( A \) can derive \( B \) out. In this whole article, symbol \( \lfloor x \rfloor \) denotes the floor function, an integer function of the real number \( x \) such that \( x - 1 < \lfloor x \rfloor \leq x \) or equivalently \( \lfloor x \rfloor \leq x < \lfloor x \rfloor + 1 \). Symbol \( a|b \) means \( b \) can be divided by \( a \); symbol \( (a, b) \) is to express the Greatest Common Divisor (GCD) of integers \( a \) and \( b \). A tracing step or a searching step is the computation of a father based on a son or vice versa.

**Lemmas**

**Lemma 1 (Node Calculation, see in (Wang, 2016a))**

Node \( N_{k,j,d} \) of \( T_3 \) is calculated by:
\[
N_{k,j,d} = 2^{d+1} + 1 + 2j
\]
\( k = 0, 1, 2, \ldots; j = 0, 1, \ldots, 2^{d} - 1 \)

Node \( N_{k,j}^* \) of \( T_X \) is computed by:
\[
N_{k,j}^* = 2^k x - 2^k + 2j + 1
\]
\( k = 0, 1, 2, \ldots; j = 0, 1, \ldots, 2^{d} - 1 \)

**Lemma 2 (Divisors on Borders, see in (Wang and Guo, 2019))**

Let \( p \) be an odd integer and \( T_p \) be the \( p \)-rooted valued binary tree and \( d \) be a positive integer with \( 1 \leq d \leq p-1 \); if there exits a positive integer \( e \) such that \( 1 \leq e \leq 2^{d-1}-1 \) and \( 2^d - (2d - 1) \equiv 0 \pmod{p} \), then \( p | N_{d,-d}^p \); if there exits a positive integer \( f \) such that \( 0 \leq f \leq f \leq 2^{d-1}-2 \) and \( 2^d + (2f - 1) \equiv 0 \pmod{p} \), then \( p | N_{d,p}^p \).

Particularly, if \( 2d+1 \equiv 0 \pmod{p} \) then \( N_{d,0}^p \equiv 0 \pmod{p} \); if \( 2d+1 \equiv 0 \pmod{p} \) then \( N_{d,-d}^p \equiv 0 \pmod{p} \).

**Lemma 3 (Floor Function, see in (Wang, 2019))**

Properties of the floor functions with real numbers \( x \) and \( y \) and integers \( n \):

- \( (P1) \) \( \lfloor x \rfloor + \lfloor y \rfloor \leq \lfloor x + y \rfloor \leq \lfloor x \rfloor + \lfloor y \rfloor + 1 \)
- \( (P2) \) \( \lfloor x \rfloor - \lfloor y \rfloor - 1 \leq \lfloor x + y \rfloor \leq \lfloor x \rfloor - \lfloor y \rfloor + 1 \)
- \( (P13) \) \( x \leq y \Rightarrow \lfloor x \rfloor \leq \lfloor y \rfloor \)
- \( (P32) \) \( n \lfloor x \rfloor \leq n \lfloor x \rfloor + n(x) + 1 \). Taking \( n = 2 \) yields \( 2\lfloor x \rfloor \leq 2\lfloor x \rfloor + 2 \lfloor x \rfloor + 1 \)

**Main Results and Proofs**

**Theorem 1**

Let \( p > 1 \) be an odd integer and \( \alpha \) be a positive integer; if \( p < 2^\alpha + 1 \) then it holds:
\[
2^{\alpha-1} < 2^{\alpha-1} + \frac{p - 1}{2} \leq 2^\alpha - 1
\]
\[
0 \leq 2^{\alpha-1} - \frac{p + 1}{2} \leq 2^{\alpha-1} - 1
\]

whereas if \( p < 2^\alpha - 1 \) it holds:
\[
2^{\alpha-1} < 2^{\alpha-1} + \frac{p - 1}{2} < 2^\alpha - 1
\]
\[
0 < 2^{\alpha-1} - \frac{p + 1}{2} \leq 2^{\alpha-1} - 1
\]

**Proof**

See the following deductions:
\[
p < 2^\alpha + 1 \Rightarrow \frac{p - 1}{2} < 2^{\alpha-1}
\]
\[
2^{\alpha-1} < 2^{\alpha-1} + \frac{p - 1}{2} \leq 2^\alpha
\]
\[
\Rightarrow 2^{\alpha-1} < 2^{\alpha-1} + \frac{p - 1}{2} \leq 2^\alpha - 1
\]
Let $N = pq$ with $1 < p \leq q$ being odd integers; then $\lfloor \log_2 N \rfloor \geq \max(2\lfloor \log_2 p \rfloor, \lfloor \log_2 q \rfloor)$.

**Proof**

Without loss of generality, assume $1 < p \leq \sqrt{N} \leq q$. Then By Lemma 3 (P13) and (P32):

$$\log_2 N > \log_2 q \Rightarrow \lfloor \log_2 N \rfloor \geq \lfloor \log_2 q \rfloor$$

and:

$$\log_2 N \geq 2\log_2 p \Rightarrow \lfloor \log_2 N \rfloor \geq 2\lfloor \log_2 p \rfloor \geq 2\lfloor \log_2 p \rfloor$$

Hence it holds:

$$\lfloor \log_2 N \rfloor \geq \max(2\lfloor \log_2 p \rfloor, \lfloor \log_2 q \rfloor)$$

**Corollary 1**

Suppose $p$ and $q$ are odd integers with $1 < p < q$; then $N = pq$ can be factorized in $\lfloor \log_2 N \rfloor + 1$ searching steps if one of $p$ and $q$ is in the form $2^a + 1$ or $2^a - 1$ with $a$ being a positive integer.

**Proof**

According to the given conditions, there are 4 cases, $q = 2^a + 1$, $q = 2^a - 1$, $p = 2^a + 1$ and $p = 2^a - 1$, to be considered.

Consider the first case $q = 2^a + 1$; then $N = 2^a p + p$. Rewrite this by:

$$N = 2^a p - 2^a + 2^a p = 2^a p - 2^a + 2 \left(2^{a-1} + \frac{p-1}{2}\right) + 1$$

Referring to Lemma 1 and Theorem 1, it yields:

$$N = N^p_{\lfloor a, 2^{a-1}, \frac{p-1}{2}\rfloor}$$
This implies that \( N \) is a node in the right branch of \( T_p \). Consequently, there are at most \( \alpha \) steps by tracing upwards and finding out the GCD between \( N \) and its ancestors in \( T_p \). Since \( q = 2^\alpha + 1 \), it yields:

\[
\alpha = \left\lfloor \log_2(q-1) \right\rfloor \leq \log_2 q < \left\lfloor \log_2 q \right\rfloor + 1 \tag{1}
\]

For the case \( q = 2^\alpha - 1 \), it holds \( N = 2^\alpha p - p = 2^\alpha p - 2^\alpha + 2 \left( 2^{\alpha-1} - \frac{p+1}{2} \right) + 1 \). Again referring to Theorem 1, it leads to \( N = N_r^{(x_2, p)} \). This case says \( N \) is a node in the left branch of \( T_p \).

For the case \( p = 2^\alpha + 1 \) or \( p = 2^\alpha - 1 \), by Lemma 2, it knows \( N_r^{(x_2, -1)} \equiv 0 \pmod{p} \) or \( N_r^{(x_2, 0)} \equiv 0 \pmod{p} \) respectively. Since \( \alpha \leq \left\lfloor \log_2 p \right\rfloor + 1 \), by genetic property it knows \( p \) can be found in at most \( 2 \left\lfloor \log_2 p \right\rfloor + 1 \) steps by tracing downwards and finding the GCD between \( N \) and nodes along the leftmost path or left side-path of \( T_N \).

**Example 1**

Let \( N = 527 \); then \( N \)'s ancestors are 263, 131, 65, 33 and 17, as depicted with Fig. 2. It can see that 17 is the divisor of \( 527 = 17 \times 31 \) and \( 31 = 2^5 - 1 \).

**Example 2**

Let \( N = 561 \), then \( N \)'s ancestors are 281, 141, 71, 35 and 17, as depicted with Fig. 3. It can see that 17 is the divisor of \( 561 = 17 \times 33 \) and \( 33 = 2^5 + 1 \).

**Proposition 1**

Suppose \( p \) and \( q \) are odd integers with \( 1 < p < q \); then \( N = pq \) can be factorized in \( \left\lfloor \log_2 N \right\rfloor + 1 \) searching steps if \( q \) is in either form of \( 2^\alpha - 1 \) and \( 2^\alpha + 1 \) with \( \alpha \) being a positive integer.

**Fig. 2:** The ancestors of \( N = 527 \) in \( T_{17} \)

**Fig. 3:** The ancestors of \( N = 561 \) in \( T_{17} \)
**Example 3**

Figure 4, symmetric divisors distributed in a tree are again exhibited with \( q \) in the form \( 2^\alpha - 1 \) or \( 2^\alpha + 1 \).

**Example 4**

Let \( N = 731 \); then the left side-path of \( T_{731} \) is 1459, 2919, 5839 and 11679, as depicted in Fig. 5. It can see GCD(11679,731) = 17. Likewise, the right side path is 1465, 2929, 5857 and 11713, among which it fits GCD(11713,731) = 17.

**Corollary 2**

Let \( p \) and \( q \) be odd integers with \( 1 < p < q \) and suppose \( q = 2^u p + 1 \) with \( u \geq 1 \) being an odd integer, \( \alpha \) being a positive integer and \( 1 < p < 2^\alpha + 1 \); then \( N = pq \) can be factorized in \( \lceil \log_2 p \rceil + 1 \) searching steps.

**Proof**

The condition \( q = 2^u p + 1 \) leads to:

\[
N = (2^u p + 1) p = 2^u p - 2^\alpha + 2^\alpha + p = 2^u p - 2^\alpha + 2^{\alpha - 1} + \frac{p - 1}{2} + 1
\]

Since \( 1 < p < 2^\alpha + 1 \), it knows by 1, \( 2^{\alpha - 1} < 2^{\alpha + 1} + \frac{p - 1}{2} \leq 2^\alpha - 1 \). Thereby:

\[
N = N^{\left(\frac{p - 1}{2}\right)}_{2^u, p - 1}
\]

This says that \( N \) is a node in the right branch of \( T_{up} \). Thus there are at most \( \alpha \) searching steps to trace upwards and find out the GCD between \( N \) and its ancestors in \( T_{up} \).
Example 5

Let $N = 6707$; then $N$'s ancestors are $3353$, $1677$, $839$, $419$, $209$, among which GCD$(6707, 209) = 19$, which results in $6707 = 19 \times 353 = 19 \times (2^5 \times 11 + 1)$. Figure 6 shows the tracing path from $6707$ to $209$. Seen from the figure, $N = 6707$ is sure in the right branch of $T_{209}$.

Corollary 3

Let $p$ and $q$ be odd integers with $1 < p < q$ and suppose $q = 2^\alpha u - 1$ with $u \geq 1$ being an old integer, $\alpha$ being a positive integer and $1 < p < 2^\alpha + 1$; then $N = pq$ can be factorized in $\lceil \log_2 p \rceil + 1$ searching steps.

Proof

By Theorem 1, the condition $1 < p < 2^\alpha + 1$ leads to $0 \leq 2^{\alpha - 1} \cdot \frac{p + 1}{2} \leq 2^{\alpha - 1} - 1$. Considering:

$$N = (2^\alpha u - 1)p = 2^\alpha up - 2^\alpha + 2^\alpha - p$$

$$= 2^\alpha up - 2^\alpha + 2^\alpha - 2^\alpha - \frac{p + 1}{2} + 1$$

it knows:

$$N = N^w_{u,2^{\alpha - 1} - \frac{p + 1}{2}}$$

This says that $N$ is a node in the left branch of $T_{up}$. Thus there are at most $\alpha$ searching steps to trace upwards and find out the GCD between $N$ and its ancestors in $T_{up}$.

Example 6

Let $N = 45601$; then $N$’s ancestors are $22801$, $11401$, $5701$, $2851$, $713$, among which GCD$(45601, 713) = 31$, which results in $45601 = 31 \times 1471 = 31 \times (2^6 \times 23 + 1)$. Figure 7 shows the tracing path from $45601$ to $713$. Seen from the figure, $N = 45601$ is sure in the left branch of $T_{713}$.

Corollary 4

Let $p$ and $q$ be odd integers with $1 < p < q$ and suppose $q = 2^\alpha u + 1$ with $u \geq 1$ being an old integer, $\alpha$ being a positive integer and $1 < p < 2^\alpha - 1$; then $N = pq$ can be factorized in $\lceil \log_2 p \rceil + 1$ searching steps.

Proof

By Theorem 1, the condition $1 < p < 2^\alpha - 1$ leads to $2^{\alpha - 1} < 2^{\alpha - 1} + \frac{p + 1}{2} < 2^{\alpha - 1}$. Since:

$$N = (2^\alpha u + 1)p = 2^\alpha up - 2^\alpha + 2^\alpha + p$$

$$= 2^\alpha up - 2^\alpha + 2^\alpha + \frac{p - 1}{2} + 1$$

$$= N^w_{u,2^{\alpha - 1} - \frac{p + 1}{2}}$$

it knows that $N$ is a node in the right branch of $T_{up}$. Thus there are at most $\alpha$ searching steps to trace upwards and find out the GCD between $N$ and its ancestors in $T_{up}$.

Example 7

Let $N = 42711$; then $N$’s ancestors are $21355$, $10677$, $5339$, $2669$, $1335$, $667$, among which GCD$(42711, 667) = 23$, which results in $42711 = 23 \times 1857 = 23 \times (2^6 \times 29 + 1)$. Figure 8 shows the tracing path from $42711$ to $667$. Seen from the figure, $N = 42711$ is sure in the right branch of $T_{667}$.
**Corollary 5**

Let \( p \) and \( q \) be odd integers with \( 1 < p < q \) and suppose \( q = 2^\alpha u - 1 \) with \( u \geq 1 \) being an old integer, \( \alpha \) being an positive integer and \( 1 < p < 2^\alpha \); then \( N = pq \) can be factorized in \( \lfloor \log_2 N \rfloor + 1 \) searching steps.

**Proof**

By Theorem 1, the condition \( 1 < p < 2^\alpha - 1 \) leads to \( 0 < 2^{\alpha - 1} \cdot \frac{p + 1}{2} < 2^\alpha - 1 \). Since:

\[
N = (2^\alpha u - 1)p = 2^\alpha up - 2^\alpha + 2^\alpha - p
\]

\[
= 2^\alpha up - 2^\alpha + 2^{\alpha - 1} - \frac{p + 1}{2} + 1
\]

it knows that \( N \) is a node in the left branch of \( T_{up} \). Thus there are at most \( \alpha \) searching steps to trace upwards and find out the GCD between \( N \) and its ancestors in \( T_{up} \).

**Example 8**

Let \( N = 383031 \); then \( N \)'s ancestors are 191515, 95757, 47879, 23939, 11969, 5985 and 2993, among which GCD(383031, 2993) = 73, which results in Figure 9 shows the tracing path from 383031 to 2993. Seen from the figure, \( N = 383031 \) is sure in the left branch of \( T_{2993} \).

![Fig. 9: Tracing ancestors of 383031](image)

**Theorem 4**

Let \( N = pq \) be an odd integer with \( p \) and \( q \) being odd integers and \( 1 < p < q \); suppose \( q = 2^\alpha u + 1 \) with \( u \geq 1 \) being an old integer, and \( 1 < p \leq 2^\alpha + 1 \); then \( N \) can be factorized in \( \lfloor \log_2 N \rfloor \) steps or in \( O((\log_2 N)^4) \) bit operations.

**Proof**

Let \( J_1 = 2^{\alpha + 1} - \frac{p + 1}{2} \) and \( J_2 = 2^{\alpha + 1} + \frac{p - 1}{2} \); summarizing Corollaries 1 to 5 yields Table 1.

Seen from the table and referring to the Corollaries 1 to 5, it knows the theorem holds considering it needs \( O((\log_2 N)^4) \) bit operations in computation of the GCD at each step.

**Corollary 6**

Let \( N = pq \) be an odd integer with \( p \) and \( q \) being odd integers and \( 1 < p < q \); suppose \( q = 2^\alpha u - 1 \) with \( u \geq 1 \) being an old integer and \( \alpha \) being an positive integer; if \( 2^\alpha + 1 < p \leq 2^{\alpha + 1} \) then \( N \) can be factorized in \( \lfloor \log_2 N \rfloor + 1 \) searching steps.

**Proof**

Direction calculation yields:

\[
N = (2^\alpha u - 1)p = 2^\alpha up - p
\]

\[
= 2^\alpha (up - 2) - 2^\alpha + 2^{\alpha - 1} - \frac{p + 1}{2} + 1
\]

\[
= 2^\alpha ((up - 2) - 1) + 2^{\alpha - 1} - \frac{p + 1}{2} + 1
\]

Let \( n = up - 2 \); by Theorem 2, \( 2^{\alpha + 1} \leq 2^\alpha + 2^{\alpha - 1} - \frac{p + 1}{2} < 2^\alpha - 1 \); consequently:

\[
N = N^{p \left( a, 2^\alpha + 2^{\alpha - 1} - \frac{p + 1}{2} \right)}
\]

That is to say, tracing upwards from \( N \) by \( \alpha \) steps will reach \( n \), the node left to \( up \); then:

\[
p = \text{GCD}(n + 2, N)
\]

The relations described in Corollary 6 among \( n, N \) and \( up \) are illustrated in Figure 10.

**Corollary 7**

Let \( N = pq \) be an odd integer with \( p \) and \( q \) being odd integers and \( 1 < p < q \); suppose \( q = 2^\alpha u + 1 \) with \( u \geq 1 \) being an old integer and \( \alpha \) being an positive integer; if \( 2^\alpha + 1 < p \leq 2^{\alpha + 1} \) then \( N \) can be factorized in \( \lfloor \log_2 N \rfloor + 1 \) searching steps.
Table 1: Summarized cases from Corollaries 1 to 5

| q   | p       | J           | N                                                                 |
|-----|---------|-------------|-------------------------------------------------------------------|
| q = 2^u-1 | 1 < p ≤ 2^u-1 | 0 < 2^u-1, \( \frac{p+1}{2} \) < 2^u-1 | \( N = N^{n_u} \) \( \{u, 2^u - \frac{p+1}{2}\} \) \( N \in l(T_{up}) \) |
|     | 1 < p ≤ 2^u+1 | 0 < 2^u-1, \( \frac{p+1}{2} \) ≤ 2^u-1 |                                                                   |
| q = 2^u+1 | 1 < p ≤ 2^u-1 | 2^u-1 < 2^u-1, \( \frac{p-1}{2} \) < 2^u-1 | \( N = N^{n_u} \) \( \{u, 2^u + \frac{p-1}{2}\} \) \( N \in r(T_{up}) \) |
|     | 1 < p ≤ 2^u+1 | 2^u-1 < 2^u-1, \( \frac{p-1}{2} \) ≤ 2^u-1 |                                                                   |

**Proof**

Direction calculation yields:

\[
N = \left(2^u + 1\right)p = 2^up + p
\]

\[
= 2^up + 2^{u+1} - 2^u - 2^{u+1} + 2^u + p
\]

\[
= 2^up + 2^{u+1} - 2^u + 2\left(2^{u+1} - 2^u + \frac{p-1}{2}\right) + 1
\]

\[
= 2^up + 2^{u+1} - 2^u + 2\left(\left(2^u + 2\right) - 1\right) + 2\left(2^{u+1} - 2^u + \frac{p-1}{2}\right) + 1
\]

Let \( n = up + 2 \); by Theorem 2, \( 0 < 2^{u-1} \cdot 2^u + \frac{p-1}{2} \leq 2^{u-1} - 1 \); consequently:

\[
N = N^{n_u} \left\{u, 2^u + 2^{u+1} - \frac{p+1}{2}\right\}
\]

That is to say, tracing upwards from \( N \) by \( \alpha \) steps will reach \( n \), the node left to \( up \); then:

\[
p = GCD(n-2, N)
\]

The relations described in Corollary 7 among \( n \), \( N \) and \( up \) are illustrated in Fig. 11.

**Theorem 5**

Let \( N = pq \) be an odd integer with \( p \) and \( q \) being odd integers and \( 1 < p < q \); suppose \( q = 2^u \pm 1 \) with \( u \) being an odd integer and \( \alpha \) being an positive integer; if \( 2^u + 1 < p < 2^u + 1 \) then \( N \) can be factorized in \( 3l_{\log_2N} \) +1 searching steps or in \( O(\log^2N) \) bit operations.

**Proof**

Summarizing Corollaries 6 and 7 yields to Table 2.

Table 2 shows that, \( N \) is a node of \( T_{up+2} \) or \( T_{up-2} \). Hence it easy to trace upwards from \( N \) to \( up + 2 \) or \( up - 2 \) and then find out the divisor \( p \). The time complexity is demonstrated in section 4.1.
Table 2: Summarized cases from Corollaries 6 and 7

| p | q | N |
|---|---|---|
| $2^a + 1 < p < 2^{a+1} - 1$ | $q = 2^a u - 1$ | $2^{a-1} < 2^{a-1} + \frac{p+1}{2} < 2^a - 1$ |
| | | $N = N^{p-1} \left( \frac{2^a - q^2 - r - 1}{2} \right)$ |
| | | $N \in \mathcal{r}(T_{q^2 - 1})$ |

$q = 2^a u - 1$ | $0 < 2^{a-1} - 2^a + \frac{p-1}{2} \leq 2^{a-1} - 1$ | $N = N^{p+1} \left( \frac{p-1 - q^2}{2} \right)$ |
| | | $N \in \mathcal{r}(T_{q^2 + 1})$ |

Table 3: Ten factorized samples

| Odd Integers | Factorization |
|--------------|--------------|
| 34639739     | 8191×4229    |
| 1159842729   | 131071×8849  |
| 10581684521  | 524287×20183 |
| 60782931320919664123 | 59649589127497217×1019 |
| 10263855667940024299 | 1256132134125569×8171 |
| 11527139783601774304441 | 2305843009213693951×49991 |
| 174538042279885450969073 | 266384887715241313×65521 |
| 9445156115384718744461691 | 3603109844542291969×262139 |
| 2732669846011417649053579 | 167988556341760475137×16267 |
| 505767294987463733694209 | 18446744073709551617×274177 |

Algorithm and Numerical Experiments

Algorithm

Theorems 4 and 5 provide an approach to factorize rapidly a composite odd integer $N = pq$ if $q$ is in the form $q = 2^a u \pm 1$ and $p$ satisfies $1 < p < 2^a \pm 1$ or $2^a + 1 < p < 2^a + 1$. This section presents a factoring algorithm. The whole procedure includes two subroutines and a main routine as follows.

Algorithm 1 Father (Calculate the father of a node)

1: Input Parameters: $Son$;
2: Begin
3: if $Son = 1 \pmod{4}$ then
4: return $(Son-1)/2$;
5: else
6: return $(Son + 1)/2$;
7: end if
8: End

The main routine shows, it requires at most $\lfloor \log_2 N \rfloor + 1$ searching steps to factorize $N$. Since at each searching step, it needs $O((\log_2 N)^3)$ bit operations to compute the GCD, it knows that the total computation can be completed in $O((\log_2 N)^3)$ bit operations.

Numerical Experiments with Maple 15

With the algorithm, programs in Maple are designed as list in the appendix. With the programs, ten odd integers are factorized in milliseconds in Maple. The ten numbers are list in Table 3. The biggest one is a 25 decimal-digit number 505767294987463733694209.

Algorithm 2 gcdOnBorder

1: Comment: Calculate GCD along left border
2: Input Parameters: $N$, $k$;
3: Begin
4: for $i = 1$ to $k$ do
5: Calculate $X = 2(N-1)+1$;
6: Calculate $g_x = gcd(N, X)$;
7: if $(g_x > 1)$ then
8: return $g_x$;
9: end if
10: Calculate $Y = 2(N-1)-1$;
11: Calculate $g_y = gcd(N, Y)$;
12: if $(g_y > 1)$ then
13: return $g_y$;
14: end if
15: end for
16: End

Conclusion and Future Work

Looking through the theorems and corollaries proved in previous sections, one can easily know that, for an odd composite integer $N = pq$ with $q$ being in the form of $2^a u \pm 1$ and $p$ satisfying $1 < p < 2^a \pm 1$ or $2^a + 1 < p < 2^a + 1$, it is easy to factorize $N$ with the help of the valued binary tree $T_N$. Actually, the factorization can be completed by just tracing and finding in $T_N$ the GCD between $N$ and $N$’s ancestors or between $N$ and the leftmost path $p^N_N$ as well as the left side-path $p^N_i$. Since there are a lot of odd positive integers that fit the conditions, this paper surely solves part of the problem on factoring big odd integers.

Meanwhile, readers can see from the list of bibliographies and their related references that, the tree method is in deed a valid method to study integers. This
leads to the future work. Hope more gougers join the study and solve the hard problem of integer factorization.

Acknowledgement

The research is supported by the Open Project Program of the State Key Lab of CAD&CG (Grant No. A2002) and by Foshan University and Foshan Bureau of Science and Technology under project that constructs Guangdong Engineering Center of Information Security for Intelligent Manufacturing System.

Author’s Contributions

Prof. Xingbo WANG contributes 95% of the work in this paper, including discovering and proving the corollaries and theorems as well as designing the algorithm. Mr. Junjian ZHONG contributes 5% of the work, mainly programs and does numerical experiments.

Ethics

The authors declare that there is no conflict of interests regarding the publication of this article.

References

Calik, P., Yilgorn P., Ayhanb P. and Demir A.S., 2004. Oxygen transfer effects on recombinant benzaldehydelyase production. Chem. Eng. Sci., 59: 5075-5083. DOI: 10.1016/j.ces.2004.07.070

Phulachand, K.S., 2016. An overview of cryptography. Innovat. IT, 3: 8-11.

Sarnaik, S., Gadekar D. and Gaikwad U., 2016. An overview to integer factorization and RSA in cryptography. Int. J. Adv. Res. Eng. Thechnol., 2: 21-26.

Wang, X., 2016a. Valuated binary tree: A new approach in study of integers. Int. J. Scientific Innovat. Math. Res., 4: 63-67. DOI: 10.20431/2347-3142.0403008

Wang, X., 2016b. Amusing properties of odd numbers derived from valuated binary tree. IOSR J. Math., 12: 53-57.

Wang, X., 2017a. Two more symmetric properties of odd numbers. IOSR J. Math., 13: 37-40. DOI: 10.9790/5728-1303023740

Wang, X., 2017b. Genetic traits of odd numbers with applications in factorization of integers. Global J. Pure Applied Math., 13: 493-517.

Wang, X., 2018. T3 tree and its traits in understanding integers. Adv. Pure Math., 8: 494-507. DOI: 10.4236/apm.2018.85028

Wang, X., 2019. Brief summary of frequently-used properties of the floor function: updated 2018. IOSR J. Math., 15: 30-33.

Wang, X. and Guo H., 2019. Some divisibility traits on valuated binary trees. Int. J. Applied Phys. Math., 9: 1-15. DOI: 10.17706/ijapm.2019.9.4.173-181

Yan, S.Y., 2013. Computer Number Theory and Modern Cryptography. 1st Edn., John Wiley and Sons, Singapore, ISBN-10: 1118188586. pp: 432.

Appendix: Maple Programs and Running Results

#Subroutine Father: find the father of a node
Father := proc (S)
local X, r;
  r := modp(S, 4);
  if r = 1 then X := (1/2)*S+1/2
  else X := (1/2)*S-1/2 end if;
end proc

#Subroutine gcdOnBorder
gcdOnBorder := proc (N, k)
local X, g, i;
  for i to k do
    X := 2^i*(N-1)+1;
    g := gcd(N, X);
    if 1 < g then break end if;
    X := 2^i*(N-1)-1;
    g := gcd(N, X);
    if 1 < g then break end if;
  end do;
end proc

# Main routine
doit:=proc(N)
local k, F, i, g;
  k := floor((log(N))/log(2))+1;
  g := gcdOnBorder(N, k);
  if g > 1 then return(g): fi;
  F := N;
  for i from 1 to k do
    F := Father(F);
    g := gcd(N, F);
    if g > 1 then return(g): fi;
    g := gcd(N, F-2);
    if g > 1 then return(g): fi;
    g := gcd(N, F+2);
    if g > 1 then return(g): fi;
  od;
end proc

#tested numbers
ob := Array([34639739, 1159847279, 10581684521, 10263855667940024299, 60782931320919664123, 115271397873601774304441, 17453804227988545096073, 944515611538471874461691, 27326984601141764903579, 505767294989746373694209]);
# test commands
for i to 10 do
  d1 := doit(ob[i]);
  d2 := ob[i]/d1;
  lprint(ob[i], d1, d2)
end do;

# Test results

| i | 34689739 | 4229 | 8191 |
|---|----------|------|------|
| 1 | 1159847279 | 8849 | 131071 |
| 2 | 1068184621 | 20183 | 524397 |
| 3 | 1098325967 | 8171 | 1256132134120569 |
| 4 | 65682901320919664213 | 1019 | 5640988127497217 |
| 5 | 115217197783601774304444 | 49991 | 2308843009921699991 |
| 6 | 174530424279885490069073 | 68521 | 260384887715143313 |
| 7 | 942156119384717446599 | 26239 | 38032004652291669 |
| 8 | 273068866602147849085579 | 16367 | 10798556041760475137 |
| 9 | 3057672966857063733589209 | 27417 | 1846874407859555617 |