The Proof of Collatz Conjecture

Hongyuan Ye (✉ ye001@hotmail.com)

Zhejiang Bitmap Intelligent Technology Co., Ltd. Jiaxing, Zhejiang 314100 China M.E. Pattern Recognition and Intelligent Control

Research Article

Keywords: Collatz conjecture, Collatz transform, Collatz even transform, Collatz odd transform, Collatz transform result sequence, Collatz transform times, Strong Collatz conjecture, Strong Collatz transform times, Non-negative integer inheritance decimal tree, Clone node, Collatz-leaf node, Collatz-leaf integer, Collatz-leaf node inheritance.

Posted Date: May 22nd, 2020

DOI: <https://doi.org/10.21203/rs.3.rs-30443/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

The Proof of Collatz Conjecture

Hongyuan Ye*

[Abstract] This paper redefines Collatz conjecture, and proposes strong Collatz conjecture, the strong Collatz conjecture is a sufficient condition for the Collatz conjecture. Based on the computer data structure—tree, we construct the non-negative integer inheritance decimal tree. The nodes on the decimal tree correspond to non-negative integers. We further define the Collatz-leaf node (corresponding to the Collatz-leaf integer) on the decimal tree. The Collatz-leaf nodes satisfy strong Collatz conjecture. Derivation through mathematics, we prove that the Collatz-leaf node (Collatz-leaf integer) has the characteristics of inheritance. With computer large numbers and big data calculation, we conclude that all nodes at depth 800 are Collatz-leaf nodes. So we prove that strong Collatz conjecture is true, the Collatz conjecture must also be true. And for any positive integer N greater than 1, the minimum number of Collatz transform times from N to 1 is $\log_2 N$, the maximum number of Collatz transform times is $800 * (N-1)$. The non-negative integer inheritance decimal tree proposed and constructed in this paper also can be used for the proof of other mathematical problems.

[Keywords] Collatz conjecture, Collatz transform, Collatz even transform, Collatz odd transform, Collatz transform result sequence, Collatz transform times, Strong Collatz conjecture, Strong Collatz transform times, Non-negative integer inheritance decimal tree, Clone node, Collatz-leaf node, Collatz-leaf integer, Collatz-leaf node inheritance.

1.Introduction

Take any positive integer greater than 1. if it's even, divide it by 2. If it's odd, multiply it by 3 and add 1. Repeat this process, the final result is 1. This is the Collatz conjecture which is quite possibly the simplest unsolved problem in mathematics.

In August 1999, mathematicians from all over the world gathered in Eichstadt, Germany, to hold a two-day international seminar on Collatz conjecture. As a special issue, American Mathematical Society released the "Ultimate Challenge: $3x + 1$ Problem."

"This is a really dangerous problem. People become obsessed with it and it really is impossible," said Jeffrey Lagarias, a mathematician at the University of Michigan and an expert on the Collatz conjecture.

In September 2019, one of the top mathematicians in the world dared to confront the problem — and came away with one of the most significant results on the Collatz conjecture in decades.

Dr. Terence Tao posted a proof showing that — at the very least — the Collatz conjecture is "almost" true for "almost" all numbers. While Tao's result is not a full proof of the conjecture, it is a major advance on a problem that doesn't give up its secrets easily.

2. Redefinition of Collatz conjecture

For the discussion, we make the following definition.

(**Definition 2.1**) **Collatz transform**: Take any positive integer N greater than 1, if N is even, divide N by 2, i.e. transform $N / 2$; if N is odd, multiply N by 3 plus 1 and divide it by 2, i.e. transform $(3N + 1) / 2$.

Collatz transform is divided into Collatz even transform and Collatz odd transform.

(**Definition 2.2**) **Collatz even transform**: Take any positive integer N greater than 1, if N is even, divide N by 2, i.e. transform $N / 2$.

(**Definition 2.3**) **Collatz odd transform**: Take any positive integer N greater than 1, if N is odd, multiply N by 3 plus 1 and divide it by 2, i.e. transform $(3N + 1) / 2$.

(**Definition 2.4**) **Collatz transform result sequence**: Take any positive integer N greater than 1, perform Collatz transforms on N , the list of integers you get as you repeat the process is defined as Collatz transform result sequence.

For example: The Collatz transform result sequence of the positive integer 8 is {8, 4, 2, 1}. The result sequence of the positive integer 11 is {11, 17, 26, 13, 20, 10, 5, 8, 4, 2, 1}.

Now we redefine the above mentioned Collatz conjecture as below:

(**Conjecture 2.5**) **Collatz conjecture**: Take any positive integer N greater than 1, perform Collatz transforms on N , repeat the process m_c times, the final result must be 1.

In the above definition, We define m_c as **Collatz transform times** (**Definition 2.6**) .

From the Collatz transform result sequences of integers 8 and 11: The Collatz transform times for the integer 8 is 3 and the Collatz transform times for the integer 11 is 10.

Suppose Collatz conjecture is correct, for a positive integer N greater than 1, if its Collatz transform result sequence are all even, the Collatz transform times m_{cMin} is the minimum, and we have:

$$N = 2^{m_{cMin}}$$

$$\text{then } m_{cMin} = \log_2 N \quad (2-1)$$

For positive integer 8, its Collatz transform result sequence are all even, so its Collatz transform times satisfies (2-1):

$$\begin{aligned} m_{cMin} &= \log_2 8 \\ &= 3 \end{aligned}$$

From (2-1), when $N \rightarrow +\infty$, the minimum number of Collatz transform times: $m_{cMin} \rightarrow +\infty$.

The expression of Collatz conjecture is quite possibly the simplest in unsolved conjecture, it's hard to find a mathematical method to solve the problem. We will seek another expression of the Collatz conjecture to find a solution

(Conjecture 2.7) Strong Collatz conjecture: Take any positive integer N greater than 1, perform Collatz transforms on N , repeat the process m_{sc} finite times, and the transform result $N_m < N$.

(Definition 2.8) Strong Collatz transform times: Take any positive integer N greater than 1, perform Collatz transforms on N , repeat the process m_{sc} finite times, and the transform result $N_m < N$; If in the processes of the first $m_{sc}-1$ times, the transform results are all greater than integer N , then the transform times m_{cs} is defined as **Strong Collatz transform times**

Below we will prove that if the strong Collatz conjecture is true, the Collatz conjecture must also be true.

According to the strong Collatz conjecture: take any positive integer N greater than 1, perform m_n finite Collatz transforms on N , and the transform result $N_m < N$. Because of $N_m < N$, we can assume that N_m takes the largest possible integer $N-1$; perform m_{n-1} finite Collatz transforms on $N-1$, and the transform result is $N-2$; and so on, we get the following Collatz transform result sequence:

$$\{N, N-1, N-2 \dots N-k+1, N-k, N-k-1, \dots 2, 1\}$$

The above sequence has N elements, its corresponding sequence of strong Collatz transform times is as below:

$$\{m_n, m_{n-1}, m_{n-2} \dots m_{n-k+1}, m_{n-k}, m_{n-k-1} \dots m_3, m_2\}$$

This sequence has $N-1$ elements. Let m_{scMax} be the maximum value of the sequence of strong Collatz transform times above.

Then the maximum number of Collatz transform times from N to 1:

$$m_{cMax} = m_n + m_{n-1} + \dots + m_{n-k+1} + m_{n-k} + m_{n-k-1} + \dots + m_3 + m_2$$

$$m_{cMax} < m_{scMax} * (N-1) \quad (2-2)$$

Therefore, if the strong Collatz conjecture is true, the Collatz conjecture must also be true; the strong Collatz conjecture is a sufficient condition for the Collatz conjecture.

3. Computer Data Structure - Tree

A tree is a data structure for which the running time of most operations is $O(\log N)$ on average. A tree can be defined in several ways. One natural way to define a tree is recursively. A tree is a collection of nodes. The collection can be empty; otherwise, a tree consists of a distinguished

node r , called the root. The remaining nodes can be divided into disjoint subsets, and each subset is itself a tree. Figure1 shows a typical tree.

Figure 1 A typical Tree

In the tree of Figure 1, the root is A. Node B has A as a parent and E, F as children. Nodes with no children are known as leaves; the leaf nodes in the tree above are K, L, F, G, M, I, J. Leaf nodes are also called termination nodes.

For any node n_i , the depth of n_i is the length of the unique path from the root to n_i . Thus, the root is at depth 0. For the tree in Figure1, B, C and D are at depth 1; E is at depth 2; K, L and M are at depth 3.

If there is a path from n_1 to n_2 , then n_1 is an ancestor of n_2 and n_2 is a descendant of n_1 . For the tree in Figure1, B is an ancestor of E, F, K and L. E, F, K and L are descendants of B.

4. Non-negative integer inheritance decimal tree

(Definition 4.1) Non-negative integer inheritance decimal tree: A tree that each node corresponds to a non-negative integer, and the depth of the node equals the number of digits of

the non-negative integer. Each node has 10 child nodes, the 10 child nodes contain all the numbers of the parent node and add one digit (0, 1, 2, 3, 4, 5, 6, 7, 8 or 9) in front of the parent node. Figure 2 shows a typical Non-negative integer inheritance decimal tree.

Figure 2 Non-negative integer inheritance decimal tree.

4.1 Basic characteristics of the non-negative integer inheritance decimal tree

- (1) The root is at depth 0, and it does not have a number.
- (2) Each node corresponds to a non-negative integer, and the depth of the node equals the number of digits of the non-negative integer.
- (3) Each node has 10 child nodes, the 10 child nodes contain all the digits of the parent node, and add one digit (0, 1, 2, 3, 4, 5, 6, 7, 8 or 9) in front of the parent node to form these 10 child nodes.
- (4) If there is a path from n_1 to n_2 , then n_1 is an ancestor of n_2 and n_2 is a descendant of n_1 . The integer corresponding to the descendant node is called descendant integer; The integer corresponding to the ancestor node is called ancestor integer.
- (5) A node with the first digit of 0 is defined as a clone node. The clone node is a direct inheritance of its parent.
- (6) There are 10 nodes at depth 1, corresponding to 10 one-bit non-negative integers (0 to 9) which includes a clone node 0.
- (7) There are 100 nodes at depth 2, corresponding to 100 two-bit non-negative integers (00 to 99) which Includes 10 clone nodes (00, 01, 02 ...09) .
- (8) There are 10^n nodes at depth n , corresponding to 10^n n -bit non-negative integers (0 to $10^n - 1$) which Includes $10^{(n-1)}$ clone nodes with the first digit of 0.

Based on the non-negative integer inheritance decimal tree, including the clone nodes, we define the following node.

(Definition 4.1) Collatz-leaf node: Take one node at depth m on the non-negative integer inheritance decimal tree where the corresponding m -bit integer N_{mx} is greater than 1. If the strong Collatz transform times of N_{mx} is less than or equal to the depth m , then this node at depth m is defined as **Collatz leaf node**, and the corresponding m -bit integer N_{mx} is defined as **Collatz- leaf integer (Definition 4.2)** .

For the decimal tree in Figure 2, at depth 1, nodes 2, 4, 6, 8 are Collatz-leaf nodes. Because the strong Collatz transform times of any positive even integer is 1, all positive even integers are Collatz- leaf integers.

For the Collatz- leaf node, we have the following important theorem.

(Theorem 4.3) Collatz-leaf node inheritance theorem: The descendant nodes of a Collatz-leaf node are also Collatz-leaf nodes; The descendant integers of a Collatz-leaf integer are also Collatz-leaf integers; The strong Collatz transform times of the descendant integer of a Collatz-leaf integer is not bigger than that of the Collatz-leaf integer.

Below we will prove the Collatz-leaf node inheritance theorem.

For the decimal tree in Figure 2, take one node Node_{mx} at depth m . Assume that this node is a Collatz-leaf node, the m -bit positive integer N_{mx} corresponding to this node must be a Collatz-leaf integer. Perform Collatz transforms on N_{mx} m times and the transform result is N_{mxc} , $N_{mxc} < N_{mx}$ must be true. Take one node Node_{nx} at depth n . Assume that this node is a descendant of node Node_{mx} and the corresponding n -bit positive integer is N_{nx} , perform Collatz transforms on N_{nx} m times and the transform result is N_{nxc} . We will prove that $N_{nxc} < N_{nx}$ must also be true.

The m -bit Collatz-leaf integer N_{mx} can be expressed as follows:

$$N_{mx} = d_m d_{m-1} \dots d_i \dots d_3 d_2 d_1$$

Where d_i is any number among 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. Because N_{nx} is a n -bit positive integer which is a descendant of N_{mx} , the low m bits of N_{nx} must be $d_m d_{m-1} \dots d_i \dots d_3 d_2 d_1$.

The n -bit positive integer N_{nx} can be expressed as follows:

$$N_{nx} = d_n d_{n-1} \dots d_{m+1} d_m d_{m-1} \dots d_3 d_2 d_1$$

Divide N_{nx} into two parts at bit m :

$$\begin{aligned} N_{nx} &= d_n d_{n-1} \dots d_{m+1} d_m d_{m-1} \dots d_3 d_2 d_1 \\ &= (d_n d_{n-1} \dots d_{m+1}) * 10^m + d_m d_{m-1} \dots d_3 d_2 d_1 \end{aligned}$$

Let the high part of the N_{nx} be N_{nm} and its low part be n_{mx} :

$$N_{nm} = (d_n d_{n-1} \dots d_{m+1}) * 10^m \quad (3-1)$$

$$N_{mx} = d_m d_{m-1} \dots d_3 d_2 d_1 \quad (3-2)$$

$$\text{then } N_{nx} = N_{nm} + N_{mx} \quad (3-3)$$

We know that the Collatz transform is divided into Collatz even transform and Collatz odd transform. Perform Collatz transforms on N_{nx} m times, suppose that the Collatz even transforms are i times, and the Collatz odd transforms are j times ($i + j = m$; $i > 0$, $j \geq 0$).

(1) If N_{nx} is an even positive integer, perform the Collatz even transforms:

$$N_{nx} / 2 = (N_{nm} + N_{mx}) / 2$$

$$= N_{nm} / 2 + N_{mx} / 2$$

From the above formula, perform the Collatz even transform on N_{nm} and N_{mx} independently, both N_{nm} and N_{mx} are divided by 2 for each Collatz even transform.

Perform the Collatz even transform on N_{nm} i times, the ratio coefficient:

$$K_{nmi}=1 / 2^i \quad (3-4)$$

Perform the Collatz even transform on N_{mx} i times, the ratio coefficient:

$$K_{mxi}=1 / 2^i \quad (3-5)$$

(2) If N_{nx} is an odd positive integer, perform the Collatz odd transforms:

$$(3 * N_{nx} + 1) / 2 = (3 * (N_{nm} + N_{mx}) + 1) / 2$$

$$\text{then } (3 * N_{nx} + 1) / 2 = (3 / 2) * N_{nm} + (3 * N_{mx} + 1) / 2 \quad (3-6)$$

From (3-6), when N_{nx} takes Collatz odd transform, N_{mx} also takes Collatz odd transform: multiply N_{mx} by 3 plus 1 and divide it by 2.

For each Collatz odd transform of N_{mx} , the ratio coefficient:

$$\begin{aligned} & ((3 * N_{mx} + 1) / 2) / N_{mx} \\ &= (3 * N_{mx} / 2 + 1 / 2) / N_{mx} \\ &= 3/2 + 1 / (2 * N_{mx}) \\ &> 3 / 2 \end{aligned}$$

Thus, perform the Collatz odd transforms on N_{mx} j times, the ratio coefficient:

$$K_{mxj} > (3 / 2)^j \quad (3-7)$$

Form (3-6), when N_{nx} takes the Collatz odd transform, N_{nm} does not take Collatz odd transform. It takes the odd transform: multiply N_{nm} by 3 and divide it by 2. So N_{nm} takes odd transform j times, the ratio coefficient:

$$K_{nmj} = (3 / 2)^j \quad (3-8)$$

(3) Based on the above analysis, N_{nx} takes the Collatz transforms m times, its high part N_{nm} also takes transforms m times, the transform result of N_{mn} is N_{mnc} :

$$N_{mnc} = K_{nmi} * K_{nmj} * N_{nm}$$

Substitute (3-4) and (3-8) into the above formula:

$$\begin{aligned} N_{nmc} &= (1/2)^i * (3/2)^j * N_{nm} \\ &= (3^j * N_{nm}) / 2^{(i+j)} \\ &= (3^j * N_{nm}) / 2^m \end{aligned}$$

$$\text{Then } N_{nmc} = (3^j / 2^m) * N_{nm} \quad (3-9)$$

Meanwhile, when N_{nx} takes the Collatz transforms m times, its low part N_{mx} also takes the Collatz transforms m times: the transform result of N_{mx} is N_{mxc} :

$$N_{mxc} = K_{mxi} * K_{mxj} * N_{mx}$$

Substitute (3-5) and (3-7) into the above formula:

$$N_{mxc} > (1/2)^i * (3/2)^j * N_{mx}$$

$$\begin{aligned} \text{Then } N_{mxc} &> (3^j / 2^m) * N_{mx} \\ N_{mxc} / N_{mx} &> 3^j / 2^m \end{aligned} \quad (3-10)$$

Because N_{mx} is a m -bit Collatz-leaf integer, N_{mx} takes Collatz transforms m times, the transform result N_{mxc} must be less than N_{mx} :

$$N_{mxc} < N_{mx} \quad (3-11)$$

$$\text{Then } N_{mxc} / N_{mx} < 1 \quad (3-12)$$

From (3-10) and (3-12), we can get:

$$(3^j / 2^m) < 1 \quad (3-13)$$

From (3-9) and (3-13), we can get:

$$N_{nmc} < N_{nm} \quad (3-14)$$

Perform the Collatz transforms on N_{nx} m time, its high part N_{nm} and low part N_{mx} take the transforms m times independently. The transform result of N_{nx} is N_{nxc} , the transform result of N_{nm} is N_{nmc} , and the transform result of N_{mx} is N_{mxc} . Then:

$$N_{nx} = N_{nm} + N_{mx} \quad (3-3)$$

$$N_{nxc} = N_{nmc} + N_{mxc} \quad (3-15)$$

Also, we have the following formulas:

$$N_{nmc} < N_{nm} \quad (3-14)$$

$$N_{mxc} < N_{mx} \quad (3-11)$$

Integrate formulas (3-3) (3-15) (3-14) and (3-11), we can get:

$$N_{nxc} < N_{nx} \quad (3-16)$$

Now we conclude: If the m -bit integer N_{mx} is a Collatz-leaf integer, its n -bit descendant positive integer N_{nx} must also be Collatz-leaf integer. Correspondingly, if a node at depth m is a Collatz-leaf node, then its descendant nodes at depth n must also be Collatz-leaf nodes.

In the above proof, the strong Collatz transform times of N_{mx} is m , and the strong Collatz transform times of N_{nx} is also m . So, the strong Collatz transform times of the descendant integer of a Collatz-leaf integer is not bigger than that of the Collatz-leaf integer.

The above proof, we suppose the following is true: Perform the Collatz transforms on N_{nx} m times, the transform result of its high part N_{nm} must be an integer.

According to the above proof: Perform Collatz transforms on N_{nx} , whether it is Collatz even transform or Collatz odd transform; N_{nm} is divided by 2 at each Collatz transform; Perform Collatz transforms on N_{nx} m times, N_{nm} should be divided by 2^m .

Form (3-1), the high part of N_{nx} :

$$\begin{aligned} N_{nm} &= (d_n d_{n-1} \dots d_{m+1}) * 10^m \\ &= (d_n d_{n-1} \dots d_{m+1}) * 5^m * 2^m \end{aligned}$$

Because N_{nm} contains 2^m , N_{nm} is divided by 2^m and the result N_{nmc} must also be an integer. This is also the reason why we set the strong Collatz transform times to be less than or equal to the depth of the node when we define the Collatz-leaf node.

4.3 In summary, we have the following conclusions:

- (1) If the Collatz conjecture is true, for any positive integer N greater than 1, the minimum number of its Collatz transform times:

$$m_{cMin} = \log_2 N \quad (2-1)$$

- (2) If the strong Collatz conjecture is true, the Collatz conjecture must also be true, and for any positive integer N greater than 1, the maximum number of its Collatz transform times:

$$m_{cMax} < m_{scMax} * (N-1) \quad (2-2)$$

m_{scMax} is a finite positive integer which is the maximum number of strong Collatz transform times.

- (3) By constructing a non-negative integer inheritance decimal tree, we defined the Collatz-leaf node on the decimal tree. Furthermore, we proposed and proved the Collatz-leaf node Inheritance theorem (Theorem4.3) .

Because of the inheritance of Collatz leaf node, with the increase of node depth, the proportion of Collatz-leaf nodes in the total nodes is also increasing. If all nodes are Collatz-leaf nodes at a limited depth, then all nodes at and below this depth are Collatz leaf nodes. Correspondingly, all positive integers at and below this depth are Collatz leaf integers. Because the clone nodes at this depth correspond to all non-negative integers less than this depth, so we prove that all integers greater than 1 satisfy the strong Collatz conjecture. The strong Collatz conjecture is true, the Collatz conjecture must also be true,

5. Computer verification and processing

In the rest of this paper, we will prove by computer calculation and processing: All nodes at and below depth 800 are Collatz-leaf nodes. Correspondingly, all the integers greater than or equal to 10^{800} are Collatz leaf integers. Because the clone nodes at depth 800 correspond to all non-negative integers less than 10^{800} . So we prove that all integers greater than 1 satisfy the strong Collatz conjecture and the maximum number of strong Collatz transform times m_{scMax} is 800.

Strong Collatz conjecture is true, the Collatz conjecture must also be true. From (2-2) , for any positive integer N greater than 1, the maximum number of Collatz transform times:

$$m_{cMax} < 800 * (N-1)$$

5.1 Collatz-leaf node inheritance verification

Collatz-leaf node inheritance is the core theorem of this paper. By computer calculation and processing, we will verify the correctness of this theorem. Traversal the nodes at depth 10 (from 0,000,000,002 to 9,999,999,999), get all Collatz-leaf nodes, for each Collatz-leaf node at depth 10, verify that its 9 child nodes at depth 11 are also Collatz-leaf nodes. Figure 3 is the computer main program to verify Collatz-leaf node inheritance theorem.

```

/// <summary>
/// CollatzLeafNodeInheritanceVerification() Collatz-leaf node inheritance verification
/// Traverse the nodes at depth 10, get all Collatz-leaf nodes at depth 10;
/// For each Collatz-leaf nodes at depth 10,
/// verify that its 9 child nodes at depth 11 are all Collatz-leaf nodes
/// </summary>
/// <param>
/// public ulong g_uSumAllNode=0;          //total nodes
/// public ulong g_uSumLeafNode=0;         //Collatz-leaf nodes
/// public ulong g_uSumInheritedNode=0;     //Collatz-leaf inheritance nodes
/// public ulong g_uNodeInteger = 0;       //the integer corresponding to the node (2 to 9999999999)
/// public bool g_bLeafNode=true;          //Collatz-leaf node
/// public bool g_bInheritedNode=true;     //Collatz-leaf inheritance node
/// </param>
1 个引用
private void CollatzLeafNodeInheritanceVerification()
{
    ulong uIndexNode = 0;          //index of node

    g_uSumAllNode = 0;              //initialize the sum of total nodes
    g_uSumLeafNode = 0;             //initialize the sum of Collatz-leaf nodes
    g_uSumInheritedNode = 0;        //initialize the sum of Collatz-leaf inheritance nodes

    //Traverse the nodes at depth 10 (2 to 9999999999)
    for (uIndexNode = 2; uIndexNode <= 9999999999; uIndexNode++)
    {
        g_uSumAllNode++;            //g_uSumAllNode++
        g_uNodeInteger = uIndexNode;

        CheckLeafNode();            /*perform Collatz transforms on g_uNodeInteger
                                     and get its strong Collatz transform times;
                                     if its strong Collatz transform times <= 10,
                                     the g_uNodeInteger is a Collatz-leaf integer
                                     set g_bLeafNode = true
                                     */

        if (g_bLeafNode == true)
        {
            g_uSumLeafNode++;        //g_uSumLeafNode++

            CheckNodeInherited();    /*Add 100000000000 ...900000000000 to g_uNodeInteger respectively
                                     Perform Collatz transforms on the 9
                                     11-bit descendants of g_uNodeInteger respectively
                                     If their strong Collatz transform times are all <= 10,
                                     the 9 11-bit integers are Collatz-leaf integers.
                                     set g_bNodeInherited = true
                                     */

            if (g_bNodeInherited == true)
            {
                g_uSumInheritedNode++; //g_uSumInheritedNode++
            }
        }
    }
    //next uIndexNode
    ShowResult();                  //Show the result;
                                   //g_uSumAllNode, g_uSumLeafNode, g_uSumInheritedNode

    return;
}

```

Figure 3 Main program to verify Collatz-leaf node inheritance

Figure 4 is the program running interface for inheritance verification. At depth 10, the total nodes are 9,999,999,998. The Collatz-leaf nodes are 9,374,999,998 and the Collatz-leaf

inheritance nodes are also 9,374,999,998. So all Collatz-leaf nodes are Collatz-leaf inheritance nodes.

Depth of Node: (Bits of Integer)	10
Start Integer:	0,000,000,002
End Integer:	9,999,999,999
Total Nodes:	9999999998
Collatz-Leaf Nodes:	9374999998
Collatz-Leaf Inheritance Nodes:	9374999998

Run Exit

Figure 4 Program running interface for inheritance verification

5.2 Strong Collatz conjecture computer verification

According to Collatz-leaf node Inheritance theorem, the descendant nodes of a Collatz-leaf node are also Collatz-leaf nodes. Therefore, we can traverse the whole non-negative integer inheritance decimal tree from the first depth, search and find the Collatz-leaf nodes on the decimal tree, take the Collatz-leaf nodes as the termination nodes with degree 0, get all the Collatz-leaf nodes and connect the neighboring Collatz-leaf nodes in turn, which will form a closed curve. The all nodes at and below the closed curve must be Collatz-leaf nodes. The depth of the lowest Collatz-leaf node on the closed curve is the maximum number of strong Collatz transform times m_{scMax} . Thus, for any positive integer N greater than 1, its maximum number of Collatz transform times:

$$m_{cMax} < m_{scMax} * (N-1)$$

Because of traversing the whole non-negative integer inheritance decimal tree, it needs Supercomputer Center and cloud computing for super large data processing. In this paper, computer program works at nodes based on depth. It traverses the nodes at depths from 1 to

12, each depth is processed separately. For the nodes below depth 12, we select some depths, and carry out random sampling calculation for each depth of nodes.

(Definition 5.1) Collatz-leaf node ratio: The ratio of Collatz-leaf node number to total node number is defined as Collatz-leaf node ratio.

Thus, when Collatz-leaf node ratio is equal to 1, all nodes at this depth are Collatz-leaf nodes.

Figure 5 is the computer main program to traverse nodes based on the depth.

```

/// <summary>
/// TraversalDepth()
/// Traversal nodes based on depth, depth from 1 to 12
/// </summary>
/// <param>
/// public ulong g_uStart = 0;           //the start integer of traversal (2)
/// public ulong g_uEnd = 0;             //the end integer of traversal (9...9)
/// public int g_nDepth = 0;             //the depth of nodes (1 to 12)
/// public ulong g_uBegin = 0;           //Begin integer of strong Collatz transform
/// public ulong g_uMid = 0;             //Middle integer of strong Collatz transform(g_uMidHigh:g_uMid)
/// public ulong g_uMidHigh = 0;         //the sum of Collatz-Leaf nodes
/// public ulong g_uSumLeafNode = 0;     //the sum of all nodes
/// public ulong g_uSumAllNode = 0;      //the Collatz-leaf node ratio
/// public double g_dRatio = 0.0;
/// </param>

private void TraversalDepth()
{
    ulong uIndexNode = 0;               //the index of node
    int nIndexTransform = 0;            //the index of strong Collatz transform

    g_uSumLeafNode = 0;                 //initialize the sum of Collatz-Leaf Nodes
    g_uSumAllNode = 0;                 //initialize the sum of All Nodes
    g_dRatio = 0.0;                    //initialize the Collatz-leaf node ratio

    for (uIndexNode = g_uStart; uIndexNode <= g_uEnd; uIndexNode++)
    {
        g_uSumAllNode++;
        g_uBegin = uIndexNode;

        //-----
        g_uMidHigh = 0;                 //Inititalize g_uMidHigh: g_uMid
        g_uMid = g_uBegin;
        for (nIndexTransform = 1; nIndexTransform < 1000; nIndexTransform++)
        {
            //-----
            if (g_uMid % 2 == 0)
            {
                EvenTransform();         //g_uMid is even: g_uMidHigh:g_uMid / 2
            }
            else
            {
                OddTransform();          //g_uMid is odd: (g_uMidHigh:g_uMid) *3 / 2
            }
            //-----
            if ((g_uMidHigh==0) & (g_uMid < g_uBegin))
            {
                if (nIndexTransform <= g_nDepth)
                {
                    g_uSumLeafNode++;    //this node is Collatz-Leaf Node
                }
                break;                  //check next node
            }
            //-----
            if (nIndexTransform > g_nDepth)
            {
                break;                  //this node is not Collatz-Leaf Node, check next node
            }
        }
        //next nIndexTransform, next transform
    }
    //next uIndexNode, check next node

    g_dRatio = (double)(g_uSumLeafNode) / (double)(g_uSumAllNode);

    ShowResult();                      //Show the result
    return;
}

```

Figure 5 Traversal nodes based on depth

Figure 6 is the program running interface for traversal nodes based on depth. The program performs traversal processes at depth 10, from 0,000,000,002 to 9,999,999,999. The total nodes are 9.999,999,998 and the Collatz-leaf nodes are 9,374,999,998, so the Collatz-leaf node ratio at tenth depth is 0.937499999.

Figure 6 Program interface of node traversal

Running the program from depth 1 to 12, the results are shown in table 1. At depth 1, the total nodes (corresponding integers greater than 1) are 8 and the Collatz-leaf nodes are 4, so the Collatz-leaf node ratio at first depth is 0.500000000. At depth 12, total nodes are 999,999,999,998 and the Collatz-leaf nodes is 944,824,218,748, so the Collatz-leaf node ratio at depth 12 is 0.944824218. From table 1, with the increase of node depth, the Collatz-leaf node ratio is also increasing.

Table 1 Collatz-leaf node ratio from depth 1 to 12

Depth of Node (Bits of Integer)	Total Nodes	Collatz-leaf nodes	Collatz-leaf node ratio
1	8	4	0.500000000
2	98	73	0.744897959
3	998	748	0.749498997
4	9,998	8,123	0.812462492

5	99,998	87,498	0.874997499
6	999,998	874,998	0.874999749
7	9,999,998	8,984,373	0.898437479
8	99,999,998	92,578,123	0.925781248
9	999,999,998	925,781,248	0.925781249
10	9,999,999,998	9,374,999,998	0.937499999
11	99,999,999,998	93,749,999,998	0.937499999
12	999,999,999,998	944,824,218,748	0.944824218

In order to process large numbers, we use 50 unsigned long integers to represent a large number. An unsigned long integer is 64 binary-bit, which can represent a maximum of 20 decimal-bit integer, and 50 unsigned long integers can represent a 1000 decimal-bit positive integer, that is, a large positive integer such as 10^{1000}

Figure 7 is the program running interface with 50 unsigned long integers. At depth 100, we randomly sample 10 billion nodes and the Collatz-leaf nodes is 9997618831, so the Collatz-leaf node ratio at depth 100 depth is 0.9997618831.

Collatz Conjecture (1000 decimal bits)

Depth of Node:
(Bits of Integer)

Total Nodes:

Collatz-leaf Nodes :

Collatz-leaf Node Ratio:

Figure 7 Program running interface with 50 unsigned long integers

On the non-negative integer inheritance decimal tree, we randomly sample 10 or 100 billion nodes at depths 50, 100, 200, 300, 400, 600 and 800, respectively. The running results are shown in table 2. At depth 800, the Collatz-leaf node ratio is 1.00000000000.

Table 2 Collatz-leaf node ratio of randomly sample

Depth of Node (Bits of Integer)	Total nodes	Collatz-leaf nodes	Collatz-leaf node ratio
50	10,000,000,000	9,967,879,308	0.9967879308
100	10,000,000,000	9,997,618,831	0.9997618831
200	10,000,000,000	9,999,941,961	0.9999941961
300	100,000,000,000	99,999,994,823	0.99999994823
400	100,000,000,000	99,999,999,976	0.99999999976
600	100,000,000,000	99,999,999,997	0.99999999997
800	100,000,000,000	100,000,000,000	1.00000000000

In order to assess the precision of the above random sampling results, we randomly sampled nodes 10 times at depth 300. Each random sample has 10 billion nodes. The Collatz-leaf node ratio is calculated separately, and the 10 results are shown in table 3.

Table 3 Collatz-leaf ratios at depth 300

No.	Total nodes (Random sampling)	Collatz-leaf nodes	Collatz-leaf node ratio
1	10,000,000,000	9,999,999,511	0.9999999511
2	10,000,000,000	9,999,999,466	0.9999999466
3	10,000,000,000	9,999,999,415	0.9999999415
4	10,000,000,000	9,999,999,429	0.9999999429
5	10,000,000,000	9,999,999,533	0.9999999533
6	10,000,000,000	9,999,999,517	0.9999999517
7	10,000,000,000	9,999,999,487	0.9999999487
8	10,000,000,000	9,999,999,443	0.9999999443
9	10,000,000,000	9,999,999,462	0.9999999462
10	10,000,000,000	9,999,999,516	0.9999999516

From table 3, among the 10 random samples, the average value of the Collatz-leaf node ratio R_{c0} is 0.9999999478. The maximum value R_{cMax} is 0.9999999533 and the minimum value R_{cMin} is 0.9999999415. The mean square error of the 10 random samples:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{10} (R_{ci} - R_{c0})^2}{10}}$$

$$= 0.0000000039$$

The mean square error is very small, so the results are quite credible.

6. Conclusions

This paper redefined Collatz conjecture. Furthermore, we proposed the strong Collatz conjecture: For any integer N greater than 1, perform the Collatz transform on N , repeat the process m_{sc} finite times, and the transform result $N_m < N$. We have proved that if strong Collatz conjecture is true, Collatz conjecture must also be true.

Based on the computer data structure - tree, a non-negative integer inheritance decimal tree is constructed, and the nodes on the inheritance decimal tree correspond to non-negative integers. We further defined the Collatz-leaf node (corresponding to the Collatz-leaf integer) on the decimal tree, Collatz-leaf nodes must satisfy strong Collatz conjecture. Derivation through mathematics, we have proved that the Collatz-leaf node (corresponding to Collatz-leaf integer) has the characteristics of inheritance. With computer large numbers and big data traversal and random sampling calculation, we have concluded that all nodes at and below depth 800 are Collatz-leaf nodes. Correspondingly, all the integers greater than or equal to 10^{800} are Collatz-leaf integers. Because the clone nodes at depth 800 correspond to all non-negative integers less than 10^{800} . So we prove that all integers greater than 1 satisfy the strong Collatz conjecture and the maximum number of strong Collatz transform times:

$$m_{scMax} < 800$$

Strong Collatz conjecture is true, Collatz conjecture must also be true. And for any integer N greater than 1, the maximum number of Collatz transform times from N to 1:

$$m_{cMax} < 800 * (N - 1)$$

7. Next steps and prospects

1. With random sampling calculation, we get the maximum number of strong Collatz transform times: $m_{scMax} < 800$. In order to calculate the determined value of the maximum strong Collatz transform times, we need Supercomputer Center and cloud computing to traverse the whole non-negative integer inheritance decimal tree.

2. The non-negative integer inheritance decimal tree proposed and constructed in this paper

also can be used for the proof of other mathematical problems. For example, the famous Goldbach conjecture: Any even number greater than 2 can be written as the sum of two prime numbers. By combining inheritance decimal tree and mathematical induction, we can use the following proof processes for the Goldbach conjecture:

(1) At depth 1 and 2, prove that any even integer greater than 2 can be written as the sum of two prime numbers.

(2) Take any even node at depth m on the inheritance decimal tree, and its corresponding m -bit decimal even integer is N_m , assuming that N_m can be written as the sum of two prime numbers.

$$N_m = P_A + P_B$$

P_A and P_B are prime numbers.

(3) Take the 9 child nodes of the above node (corresponding to N_m); that is, add one digit (1, 2, 3, 4, 5, 6, 7, 8 or 9) before N_m and get the 9 $(m+1)$ -bit even numbers: $10^m + N_m$, $2 \cdot 10^m + N_m$, $3 \cdot 10^m + N_m$, $4 \cdot 10^m + N_m$, $5 \cdot 10^m + N_m$, $6 \cdot 10^m + N_m$, $7 \cdot 10^m + N_m$, $8 \cdot 10^m + N_m$ and $9 \cdot 10^m + N_m$.

Prove that any of the 9 $(m+1)$ -bit decimal even numbers can also be written as the sum of two prime numbers.

$$N_{(m+1)} = P_{A'} + P_{B'}$$

$P_{A'}$ and $P_{B'}$ are prime numbers.

Declarations

I declare no competing interests.

References

- [1] Terence Tao, Almost all orbits of the Collatz map attain almost bounded values. arXiv:1909.03562v2 [math.PR] 13 Sep 2019
- [2] The Ultimate Challenge: the $3x+1$ problem. Edited by Jeffrey C. Lagarias. American Mathematical Society, Providence, RI, 2010.
- [3] Kevin Hartnett: <https://www.quantamagazine.org/mathematician-terence-tao-and-the-collatz-conjecture-20191211/>
- [4] M. Chamberland, A $3x+1$ survey: number theory and dynamical systems, The ultimate challenge: the $3x + 1$ problem, 57-78, Amer. Math. Soc., Providence, RI, 2010.
- [5] A. Kontorovich, J. Lagarias, Stochastic models for the $3x + 1$ and $5x + 1$ problems and related problems, The ultimate challenge: the $3x + 1$ problem, 131-188, Amer. Math. Soc., Providence, RI, 2010.
- [6] J. Lagarias, The $3x+1$ problem and its generalizations, Amer. Math. Monthly 92 (1985) 3-23.
- [7] Ivan Korec, A density estimate for the $3x + 1$ problem, Math. Slovaca 44 (1994), no. 1, 85-89.
- [8] J. Lagarias, K. Soundararajan, Benford's law for the $3x + 1$ function, J. London Math. Soc. (2) 74 (2006), no. 2, 289-303.

*Hongyuan Ye: ye001@hotmail.com

Zhejiang Bitmap Intelligent Technology Co., Ltd. Jiaxing, Zhejiang 314100 China
M.E. Pattern Recognition and Intelligent Control;

Figures

Figure 1

A typical Tree

Depth 0
Root Node

Depth 1 (10 Nodes)
10 1-bit Integers
1 Clone Node

Depth 2 (100 Nodes)
100 2-bit Integers
10 Clone Nodes

Depth m (10^m Nodes)
 10^m m -bit Integers
 10^{m-1} Clone Nodes

Depth n (10^n Nodes)
 10^n n -bit Integers
 10^{n-1} Clone Nodes

Figure 2

Non-negative integer inheritance decimal tree.

```

/// <summary>
/// CollatzLeafNodeInheritanceVerification() Collatz-leaf node inheritance verification
/// Traverse the nodes at depth 10, get all Collatz-leaf nodes at depth 10;
/// For each Collatz-leaf nodes at depth 10,
/// verify that its 9 child nodes at depth 11 are all Collatz-leaf nodes
/// </summary>
/// <param>
/// public ulong g_uSumAllNode=0;          //total nodes
/// public ulong g_uSumLeafNode=0;         //Collatz-leaf nodes
/// public ulong g_uSumInheritedNode=0;     //Collatz-leaf inheritance nodes
/// public ulong g_uNodeInteger = 0;       //the integer corresponding to the node (2 to 9999999999)
/// public bool g_bLeafNode=true;          //Collatz-leaf node
/// public bool g_bInheritedNode=true;     //Collatz-leaf inheritance node
/// </param>
1 个引用
private void CollatzLeafNodeInheritanceVerification()
{
    ulong uIndexNode = 0;          //index of node

    g_uSumAllNode = 0;              //initialize the sum of total nodes
    g_uSumLeafNode = 0;             //initialize the sum of Collatz-leaf nodes
    g_uSumInheritedNode = 0;        //initialize the sum of Collatz-leaf inheritance nodes

    //Traverse the nodes at depth 10 (2 to 9999999999 )
    for (uIndexNode = 2; uIndexNode <= 9999999999; uIndexNode++)
    {
        g_uSumAllNode++;            //g_uSumAllNode++
        g_uNodeInteger = uIndexNode;

        CheckLeafNode();            /*perform Collatz transforms on g_uNodeInteger
                                     and get its strong Collatz transform times;
                                     if its strong Collatz transform times <= 10,
                                     the g_uNodeInteger is a Collatz-leaf integer
                                     set g_bLeafNode = true
                                     */

        if (g_bLeafNode == true)
        {
            g_uSumLeafNode++;        //g_uSumLeafNode++

            CheckNodeInherited();    /*Add 10000000000 ... 90000000000 to g_uNodeInteger respectively
                                     Perform Collatz transforms on the 9
                                     11-bit descendants of g_uNodeInteger respectively
                                     If their strong Collatz transform times are all <= 10,
                                     the 9 11-bit integers are Collatz-leaf integers.
                                     set g_bNodeInherited = true
                                     */

            if (g_bNodeInherited == true)
            {
                g_uSumInheritedNode++; //g_uSumInheritedNode++
            }
        }
    }
    //next uIndexNode
    ShowResult();                  //Show the result;
                                   //g_uSumAllNode, g_uSumLeafNode, g_uSumInheritedNode

    return;
}

```

Figure 3

Main program to verify Collatz-leaf node inheritance

Collatz-Leaf Node Inheritance Verification

Depth of Node:
(Bits of Integer) 10

Start Integer: 0,000,000,002

End Integer: 9,999,999,999

Total Nodes: 9999999998

Collatz-Leaf Nodes: 9374999998

Collatz-Leaf Inheritance Nodes: 9374999998

Run Exit

Figure 4

Program running interface for inheritance verification

```

/// <summary>
/// TraversalDepth()
/// Traversal nodes based on depth, depth from 1 to 12
/// </summary>
/// <param>
///public ulong g_uStart = 0;           //the start integer of traversal (2)
///public ulong g_uEnd = 0;             //the end integer of traversal (9...9)
///public int g_nDepth = 0;             //the depth of nodes (1 to 12)
///public ulong g_uBegin = 0;           //Begin integer of strong Collatz transform
///public ulong g_uMid = 0;             //Middle integer of strong Collatz transform(g_uMidHigh:g_uMid)
///public ulong g_uMidHigh = 0;
///public ulong g_uSumLeafNode = 0;     //the sum of Collatz-Leaf nodes
///public ulong g_uSumAllNode = 0;     //the sum of all nodes
///public double g_dRatio = 0.0;       //the Collatz-leaf node ratio
/// </param>

private void TraversalDepth()
{
    ulong uIndexNode = 0;               //the index of node
    int nIndexTransform = 0;           //the index of strong Collatz transform

    g_uSumLeafNode = 0;                //initialize the sum of Collatz-Leaf Nodes
    g_uSumAllNode = 0;                 //initialize the sum of All Nodes
    g_dRatio = 0.0;                    //initialize the Collatz-leaf node ratio

    for (uIndexNode = g_uStart; uIndexNode <= g_uEnd; uIndexNode++)
    {
        g_uSumAllNode++;
        g_uBegin = uIndexNode;

        //-----
        g_uMidHigh = 0;                 //Initialize g_uMidHigh: g_uMid
        g_uMid = g_uBegin;
        for (nIndexTransform = 1; nIndexTransform < 1000; nIndexTransform++)
        {
            //-----
            if (g_uMid % 2 == 0)
            {
                EvenTransform();         //g_uMid is even: g_uMidHigh:g_uMid / 2
            }
            else
            {
                OddTransform();          //g_uMid is odd: (g_uMidHigh:g_uMid) *3 / 2
            }
            //-----
            if ((g_uMidHigh==0) & (g_uMid < g_uBegin))
            {
                if (nIndexTransform <= g_nDepth)
                {
                    g_uSumLeafNode++;    //this node is Collatz-Leaf Node
                }
                break;                  //check next node
            }
            //-----
            if (nIndexTransform > g_nDepth)
            {
                break;                  //this node is not Collatz-Leaf Node, check next node
            }
        }
        //next nIndexTransform, next transform
    }
    //next uIndexNode, check next node

    g_dRatio = (double)(g_uSumLeafNode) / (double)(g_uSumAllNode);

    ShowResult();                      //Show the result
    return;
}

```

Figure 5

Traversal nodes based on depth

Figure 6

Program interface of node traversal

Figure 7

Program running interface with 50 unsigned long integers