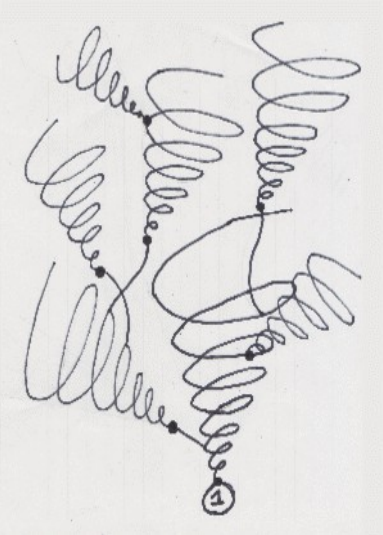
Collatz Tree Generator Manual ED:1.02

For Version 1.1.5 Release 2022-01-29



The Collatz conjecture or 3*x*+1 problem is easily understandable even by an elementary schoolboy like the following. Take an arbitrary number *n*. If *n* is even then divide it by 2 else (*n* is odd) triple it and plus 1. The conjecture states that by repeating this process any natural number *n* eventually reaches to 1.

For example let's start at 15, you get a number sequence like 15→ 46→ 23→ 70→ 35→ 106→ 53→ 160→ 80→ 40→ 20→ 10→ 5→ 16→ 8→ 4→ 2→ 1. That's it.

We believe that the solution must be as much easy as the problem itself like any elementary schoolboy has no difficulty to understand.

The Collatz graph $G(V, E)$ is a directed infinite graph such as V is the set $\mathbb{N}$ of natural numbers and E is a set of directed edge $e(i, j)$ where if i is even then $j = \frac{i}{2}$ otherwise $j = 3i+1$.

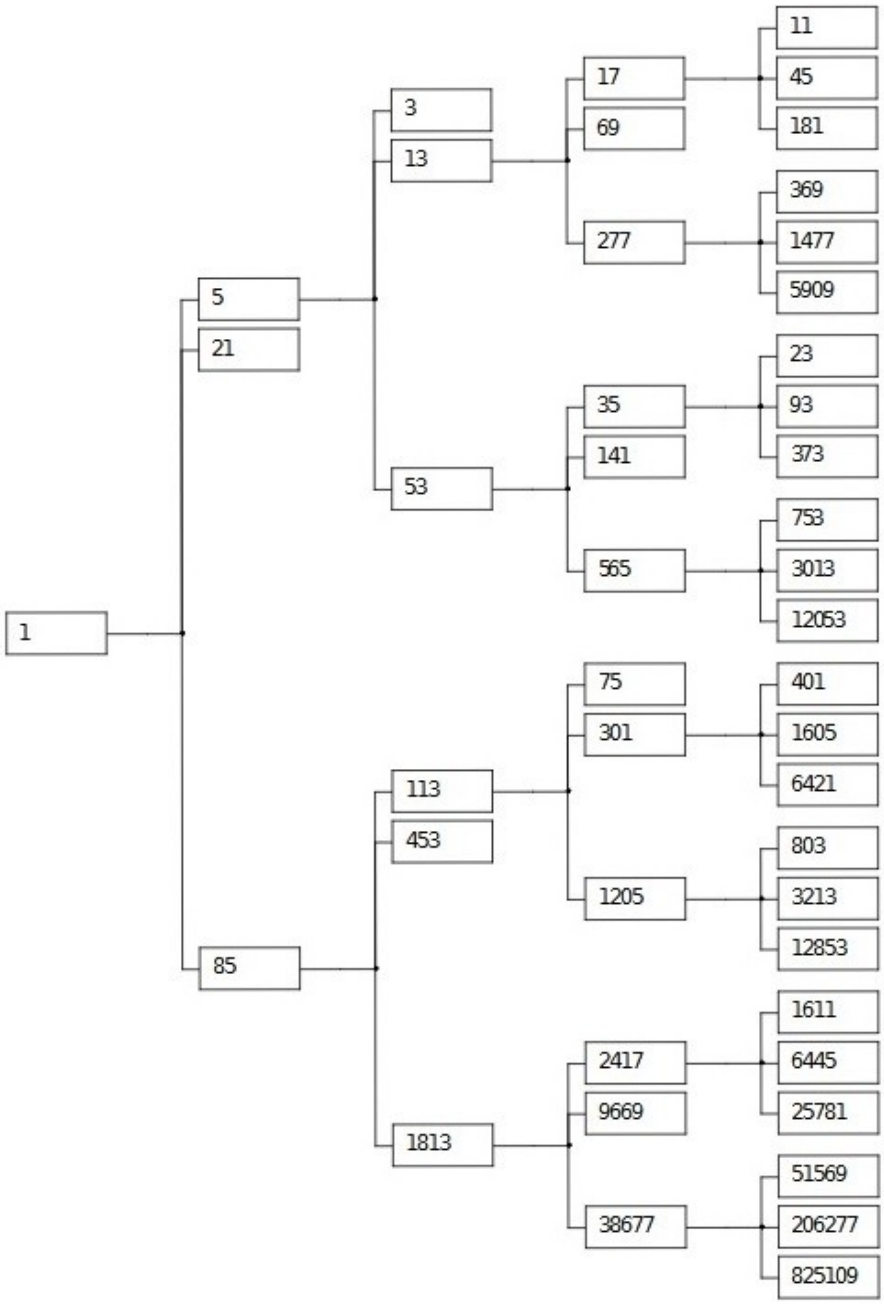
It is easy to show that as far as the Collatz graph is connected as an undirected graph it does neither circulate nor diverge other than cycle(1, 4, 2). ⇒ To see the proof of this assertion please check the appendix at the end of this manual. All through this manual we treat the Collatz graph as an undirected graph unless mentioned otherwise because our concern is just the connectivity of the graph.

In this aspect it is easy to see that every edge adjacent to an even number is neglectable because any even number *m* eventually reaches to its own odd number *n*. Contracting these edges adjacent to even numbers and merging vertices joined by them we get a graph which consists of only odd numbers. Obviously this operation does not affect the connectivity of the graph.

We call such a graph as **Collatz Tree** here. The Collatz Tree is an infinite graph with only odd numbers and the question to be asked is whether the graph is connected or not. (*To use "tree" here may be an abuse in graph theory because a tree is normally defined as a connected graph.*) The easiest way to show its connectivity is to build the whole tree from scratch starting from 1. Our Collatz Tree Generator is able to do that.

A regular tree is a tree of which nodes have the same number of branches except leaves of the tree. Collatz Tree Generator generates a regular Collatz Tree with

arbitrary degree and height. The following chart is a sample of regular Collatz Tree with root 1, degree 3 (ternary tree), height 4 and nodes 46 printed by Zelkova Tree.



The Collatz Tree is an undirected ***infinite*** graph of which vertex set is the infinite set of odd numbers. Every node in the tree except multiple of 3 has infinite number of branches. However a regular Collatz Tree is assumed to have a finite degree (branch number) and a finite height, in other words a regular Collatz Tree is a ***finite*** graph.

This program has 2 tabs and 5 functions (experiments). The first function “Collatz Tree Generator” is on tab A and the other 4 functions are on tab B. Tab A and tab B are totally independent therefore when you are running a long experiment on tab A you can do other things on tab B simultaneously.

1. **Collatz Tree Generator** generates a ***regular*** Collatz Tree of arbitrary size

(degree x height) and outputs the adjacency list of the tree in tab-separated file format

2. **Get The Sequence** outputs the Collatz number sequence and the branch order sequence of an arbitrary odd number
3. **Get the Number** outputs the odd number at an arbitrary position on the Collatz Tree designated by a branch order sequence
4. **Truncated Tree** outputs the adjacency list of a truncated Collatz subtree of which end node is designated by a branch order sequence.
5. **Verification Test** verifies the Collatz Tree structure on which the Collatz Tree Generator is entirely based by testing every odd number in a designated range.

The first function “Collatz Tree Generator” implies that the Collatz conjecture was essentially solved already. The last function “Verification Test” tests the premise. We are curious about if our solution would fail in the Verification Test. Please try these functions and make us informed with the result.

Quick Start

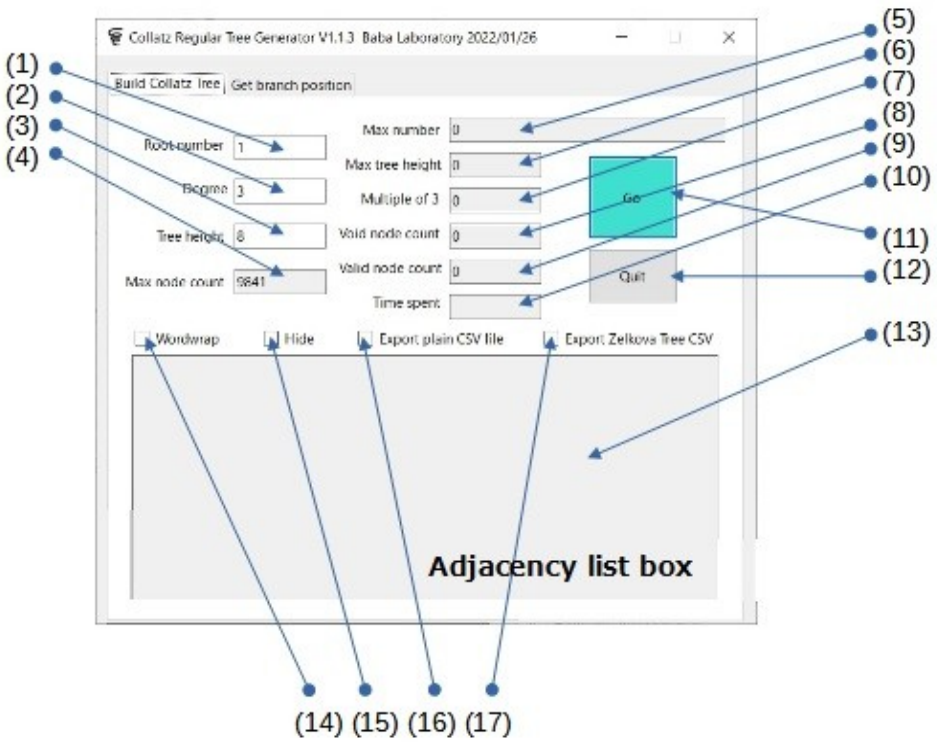
Be familiar with Collatz Tree Generator in just 3 minutes!

1. Download the program package (Collatz Tree Generator.zip) from our site.
2. Decompress the compressed *.zip file on your desktop.
3. Open the folder “Collatz Tree Generator” and double-click the CollatzTreeGenerator.exe. It does not run outside the folder and needs .NET 5.0 environment. You can download .NET 5.0 Runtime from Microsoft site.
4. On tab A (**Build Collatz Tree**) you will see a big blue button labeled as “**Go**” at the upper right corner of the window. Click it and you will see odd numbers scrolling down in the box occupying a half of the window. This is the adjacency list of a regular Collatz Tree. It takes less than one minute to show the result.
5. Click “**Get branch position**” label and go to tab B.
6. Click the big blue button labeled as “**Verify**” at the middle left side and you will see the progress of the experiment in the window. This is the Verification Test for the Collatz Tree structure. It takes less than one minute to show the result.
7. Try other three buttons, **Get the sequence**, **Get the number** and **Truncated tree** in this order and see what happens.

Tab A: Build Collatz Tree

Tab A is the main stage of this software and performs as the regular Collatz Tree Generator. The maximum degree and tree height are limited to 32767 (in the range of Int16) respectively. You need at least 2GB RAM to perform this experiment.

Screen Components on tab A



(1) ***Root number*** (input) Enter a number of the root node of a regular Collatz Tree. If the number is 1 then the output is the whole tree with designated ***degree***(2) and ***tree height***(3) otherwise it outputs a subtree of the Collatz Tree stands on the root at the designated number. The height of the subtree is measured from the root node position. ***Root number*** must be odd and not a multiple of 3 otherwise you will receive an alert message. ***Root number*** value has no limitation except out of memory case.

(2) ***Degree*** (input) Enter a number of branches of a node. Every node in a regular tree has the same number of branches except the leaves of the tree. The value of ***degree*** must be in the range of Int16 ≦32767. *This usage of "degree" may be an abuse in graph theory normally but in this manual we use it only this meaning*.

(3) ***Tree height*** (input) Enter a number of the tree height to be generated. ***Tree***

height is relative to the root node position. The value of ***tree height*** must be in the range of Int16 ≦32767.

(4) ***Max node count*** (output) The virtual node count in a regular tree with designated ***degree***(2) and ***tree height***(3). Actually any node with value of a multiple of 3 has no branches therefore valid node count of a regular Collatz Tree is much smaller than this value. It is automatically recalculated whenever ***degree*** or ***tree height*** is changed. After an experiment it must be ***max node count = void node count + valid node count.***

(5) ***Max number*** (output) The maximum node number appeared in the outputted regular Collatz Tree.

(6) ***Max tree height*** (output) The maximum height of nodes appeared in the outputted regular Collatz Tree. This number will be same with the preset ***tree height***(3) at the end of the experiment.

(7) ***Multiple of 3*** (output) The count of nodes with a value of a multiple of 3 in the outputted regular Collatz Tree. These nodes have no branches in the Collatz Tree.

(8) ***Void node count*** (output) The node count of a *complete regular tree* equals to ***max node count***(4) but actual tree has much less count of nodes. ***Void node count*** shows the count of these blank positions in the tree due to multiple of 3 nodes (leaves).

(9) ***Valid node count*** (output) The actual node count in the outputted regular Collatz Tree.

(10) ***Time spent*** (output) The time spent to build the regular tree. When the time span exceeds 24 hours it counts days.

(11) **Go** (button) Start to build a regular Collatz Tree with designated ***degree***(2) and ***tree height***(3). By clicking this button its label turns to "**Stop**" and click the "**Stop**" button terminates the session. When the task completed it turns to "**Go**" again.

(12) **Quit** (button) Quit the application. If **Go** button(11) label is "**Stop**" clicking the **Quit** button acts as the **Stop** button.

(13) **Adjacency list box** (output) Adjacency lists are outputted here. But the lines are eliminated as time goes by. You can save the whole result by checking on **Export plain CSV file** checkbox(16). Adjacency list is a commonly used way to represent a graph and the line format is such tab-separated node numbers as {parent node number, child[1] node number, ⋯, child[k] node number}.

(14) **Wordwrap** (checkbox) Fold back lines at the right end of **Adjacency list box**(13).

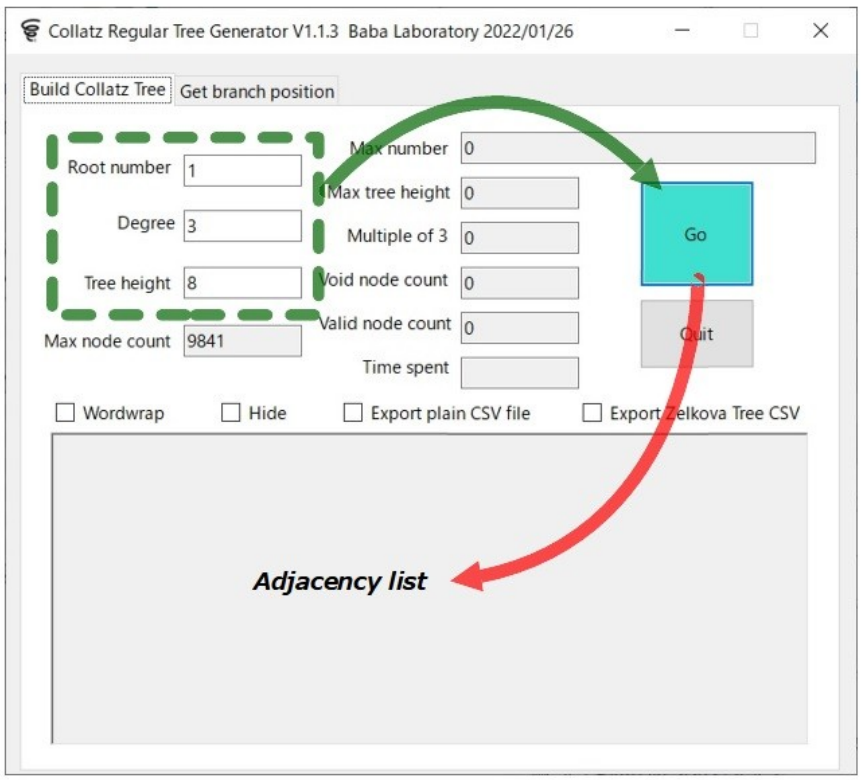
(15) **Hide** (checkbox) Hide **Adjacency list box**(13). This is effective for speeding up the progress.

(16) **Export plain CSV file** (checkbox) Designate to save the result in a tab-separated file format (CSV stands for comma separated values but usually includes tab-separated format. Both are readable with any spreadsheet software.) The file is generated on the desktop of your PC with a fixed name "***CollatzTree.csv***". This checkbox is exclusive with **Export Zelkova Tree CSV**(17).

(17) **Export Zelkova Tree CSV** (checkbox) Designate to save the result in Zelkova Tree CSV file format. Zelkova Tree is an excellent genealogy builder developed by us (Baba Laboratory). The file is generated on the desktop of your PC with a fixed name "***CollatzTree.csv***". This checkbox is effective over the functions in tab B but exclusive with **Export plain CSV file**(16).

Collatz Tree Generator

Collatz Tree Generator generates a ***regular*** Collatz Tree of arbitrary size (degree x tree height) and outputs the adjacency list of the tree in CSV file format.



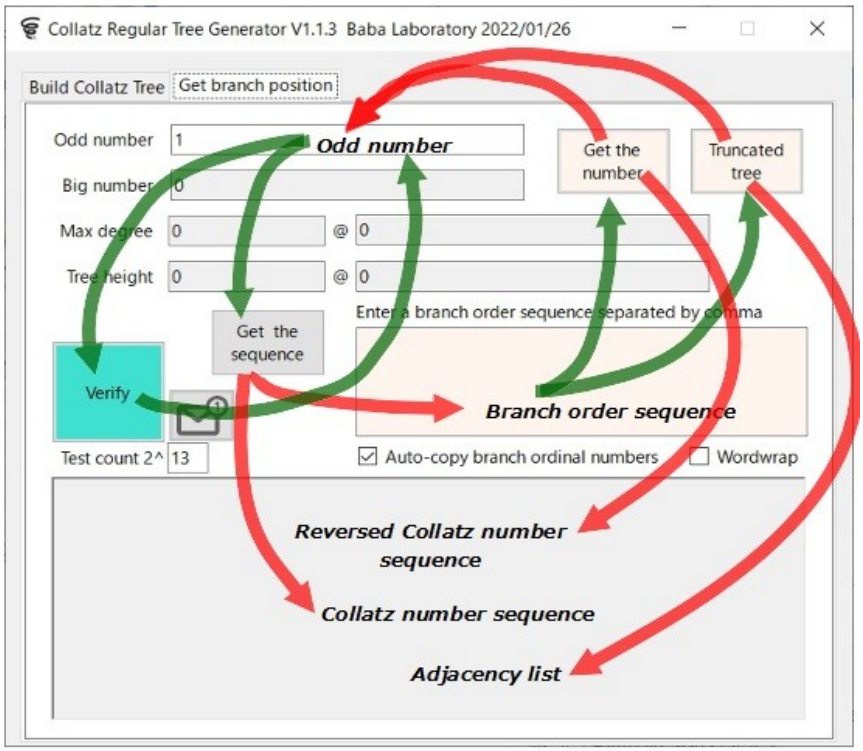
The root node of the tree is designated by ***root number***. The ***root number*** must be

odd and not a multiple of 3. Because a multiple of 3 does not have a subtree on the Collatz Tree. ⇒ See the proof of this assertion in the appendix at the end of this manual. ***Degree*** and ***tree height*** must be in the range of Int16 ≦32767.

Adjacency list is a sort of representation of graph in line format {*parent node, child node*[1]*, child node*[2] ⋯} separated by tab character. The line count of an adjacency list equals to the node count of the graph. If you uncheck **Wordwrap** then the adjacency list will be outputted as lines with no folding. You can hide the **Adjacency list box** by checking the **Hide** checkbox. This would speed up the experiment remarkably. Check **Export plain CSV file** to save the result into a CSV format file.

Tab B: Get branch position

There are 4 functions (experiments) on tab B and these 4 functions are separated into 2 groups. One is **Verification** and **Get the Sequence** and the other is **Get the number** and **Truncated tree**. The former receives ***odd number*** as an input and the latter outputs ***odd number*** as the result.



Branch order sequence is a sequence of ordinal number of branches on a Collatz Tree. This sequence is the output of **Get the sequence** while it is the input of **Get the number** and **Truncated tree**.

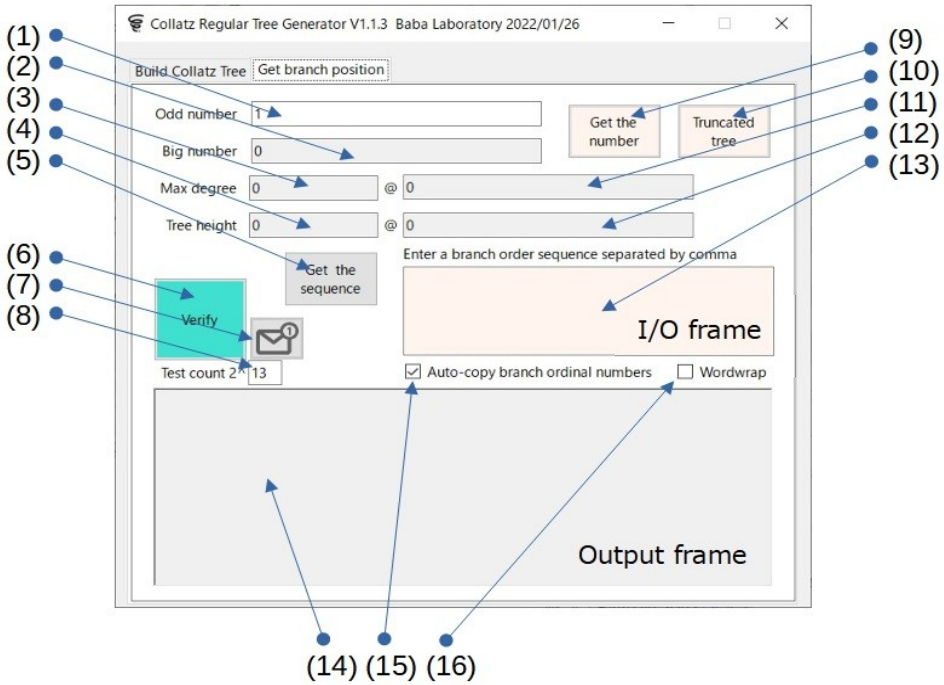
Branch order sequence uniquely decides the position of any node with an arbitrary odd number in the Collatz Tree. For example the node 6421 in the above chart (the sample regular Collatz Tree) has the branch order sequence (3, 1, 2, 3). Branch order

sequence is universal in any regular Collatz Tree with arbitrary *degree* and height provided that the root node of the tree is 1.

Collatz number sequence is a sequence of odd numbers given by applying Collatz mapping to the objective odd number. For example if the odd number is 15 then the Collatz number sequence is (*15, 23, 35, 53, 5, 1).* On the other hand ***Reversed Collatz number sequence*** is the opposite of that, i.e. (*1, 5, 53, 35, 23, 15)*.

With respect to ***adjacency list*** refer to the explanation in tab A section.

Screen Components on tab B



(1) ***Odd number*** (input/output) This field is used in dual directions depending on each experiment. The ***odd number*** must be odd otherwise you will receive an alert message. The value of the ***odd number*** has no limitation except out of memory case.

- **Get the sequence** (in) Designate a starting ***odd number*** to get its ***Collatz number sequence*** and to put it into **Output frame**(14)
- **Get the number** (out) Output the number of the terminal node of given ***branch order sequence*** in **I/O frame**(13)
- **Truncated tree** (out) Output the number of the terminal node of given ***branch order sequence*** in **I/O frame**(13)
- **Verification Test** (I/O) Designate an objective odd number *N* for a test and output the next odd number *N' = N+2* for updating ***odd number*** through

the experiment

(2) ***Big number*** (output) The biggest odd number that appeared in the experiment

(3) ***Max degree*** (output) The maximum branch order of nodes that appeared in the experiment

(4) ***Tree height*** (output) The maximum height of nodes that appeared in the experiment

(5) **Get the sequence** (button) Given ***odd number***(1), start an operation to output its ***Collatz number sequence*** in **Output frame**(14) with the branch ordinal number of the node in parenthesis. For example if the number sequence is like ...19 [2], 29 [1], 11 [2],... then it should be read like the number 19 is at the 1st branch of 29 and 29 is at the 2nd branch of 11 and so on. If the checkbox **Auto-copy branch ordinal...**(15) is checked then a branch order sequence is outputted into **I/O frame**(13).

(6) **Verify** *(button) Execute a* **Verification Test** *starting at given* ***odd number***. Verification Test verifies the Collatz Tree structure on which our **Collatz Tree Generator** is entirely based by testing every odd number in a designated range. The total test count is designated by ***test count***(8).

(7) **Send mail** (button) Send a mail for the feedback of the Verification Test result to the author of this program. The body of the mail includes ①contents in **Output frame**(14) and ②contents in **I/O frame**(13). Any other information is never transferred to the outside. You can add any additional information in **I/O frame**(13). This button is active only immediately after a **Verification Test**(6) and can be pressed just once.

(8) ***Test count*** **(input)** Designate the *Verification Test count k* by the value of a power of 2. Let the power equals to p then the test count $k=2^p$. For example if $p=5$ and the ***odd number*** (starter) equals to N then the test count $k=2^5=32$ and the next ***odd number*** $N'=N+2k=N+64$ (after the experiment). The value of p must be in the range of Int8≦127.

(9) **Get the number** (button) Given a ***branch order sequence*** in **I/O frame**, output the terminal node number of the sequence into ***odd number*** box and also output the ***Reversed Collatz number sequence*** into **Output frame** with branch ordinal numbers in parenthesis. The ***branch order sequence*** is always assumed to start at root node 1 of the Collatz Tree.

(10) **Truncated tree** (button) A truncated tree stands for a subtree of a Collatz Tree such as a chain of nodes plus leaves sticking to the stem directly. Given a

branch order sequence, output the number of the terminal node of the sequence into ***odd number*** and an adjacency list of the truncated tree into **Output frame**. The sequence is always assumed to start at the node 1.

(11) ***Max degree @*** (output) The maximum number of node at which position ***max degree*** was given

(12) ***Tree height @*** (output) The maximum number of node at which position ***tree height*** was given

(13) **I/O frame** (input/output)

- **Get the sequence** (output) Output a ***branch order sequence*** of the given odd number if the checkbox **Auto-copy branch ordinal...** is checked.
- **Get the number** (input) Enter a ***branch order sequence*** to get the terminal odd number of the sequence
- **Truncated tree** (input) Enter a ***branch order sequence*** to get the terminal odd number of the sequence
- **Verification Test** (I/O) **I/O frame** is invisible during the experiment. After the experiment this frame is opened again for users to enter their response to the author.

(14) **Output frame** (output)

- Get the sequence (output) Output a ***Collatz number sequence*** starting at the given ***odd number***
- **Get the number** (output) Output a ***Reversed Collatz number sequence*** terminating at the end node of the given branch order sequence.
- **Truncated tree** (output) Output an ***adjacency list*** of the Truncated tree of the end node of the given branch order sequence.
- **Verification Test** (output) Output the state of progress of the experiment

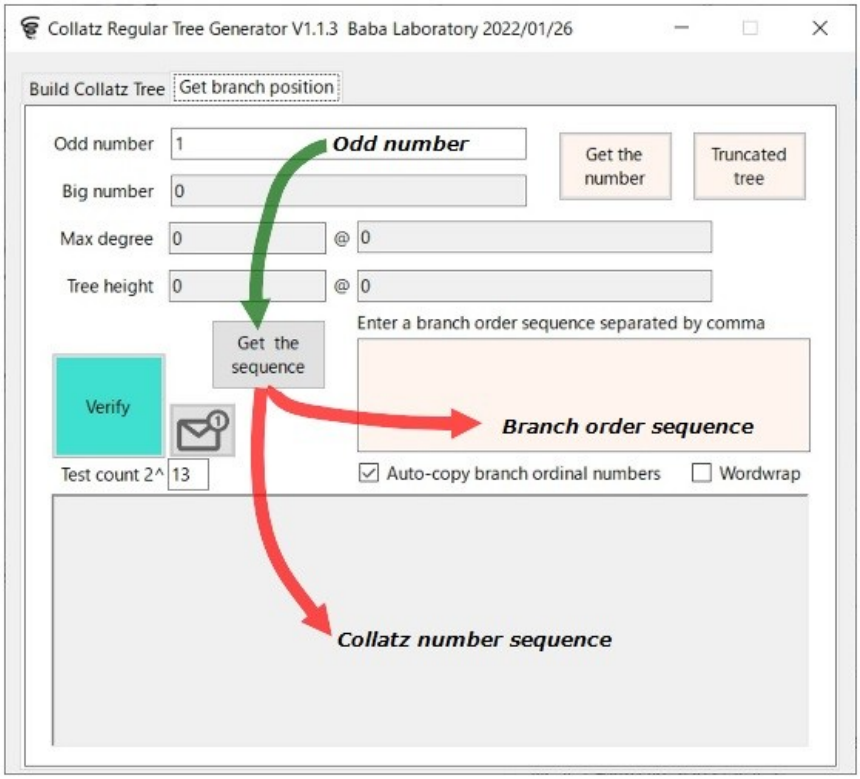
(15) **Auto-copy branch ordinal numbers** (checkbox) In a **Get the sequence** experiment obtained ***branch order sequence*** is transferred to **I/O frame**(13) in comma separated text format.

(16) **Wordwrap** (checkbox) Fold back lines at the right end of **Output frame**(14).

Get the Sequence

Get the sequence takes an odd number *N* in the ***odd number*** box as its input and outputs a ***Collatz number sequence*** in ***Output frame***. The sequence starts from the

number *N* and presumably ends at 1. As a byproduct it also outputs a ***branch order sequence*** in ***I/O frame*** provided that the checkbox **Auto-copy branch ordinal...** is ON. If the checkbox **Export Zelkova Tree CSV** is ON in **tab A** then the result is saved in "***CollatzChain.csv***" file on the desktop of your PC in Zelkova tree CSV file format.



Enter an odd number *N* (in this case 16385) in the ***odd number*** box and click **Get the sequence** button, then **Output frame** will looked like the following.

```
16385
12289 [1]
9217 [1]
6913 [1]
5185 [1]
3889 [1]
2917 [1]
547 [2]
821 [1]
77 [3]
29 [2]
11 [2]
17 [1]
13 [1]
5 [2]
1 [1]
```

Collatz number sequence is a number sequence generated by so called Collatz mapping that is defined as a function $f(n)$ for $n \in \mathbb{N}$ like,

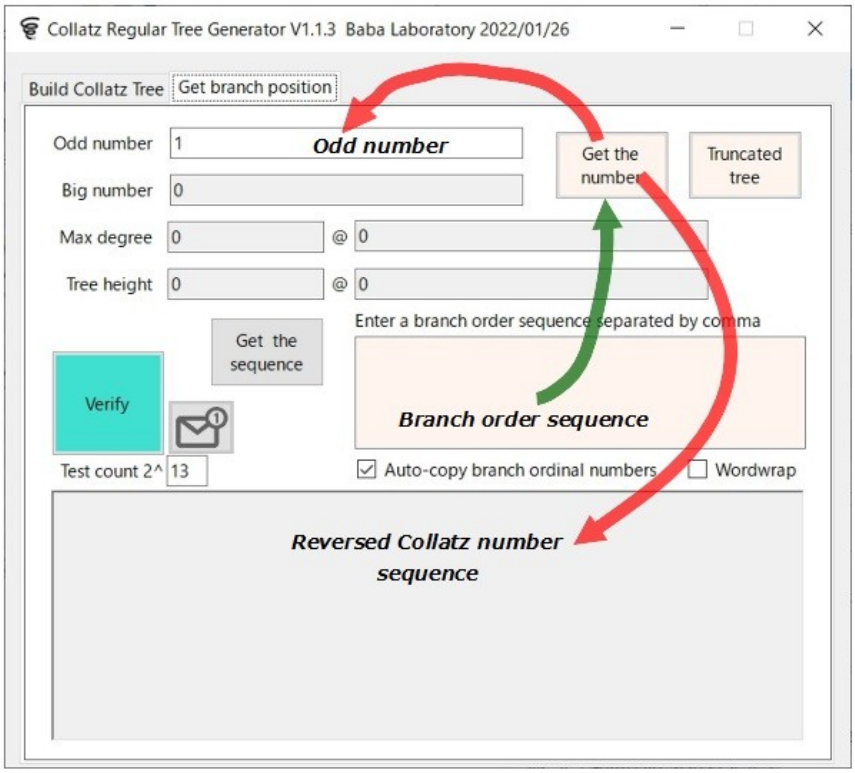
$$f(n) = \begin{cases} \frac{n}{2} & \text{if } n \equiv 0 \pmod 2 \\ 3n+1 & \text{if } n \equiv 1 \pmod 2. \end{cases}$$

However as we mentioned earlier our concern is mainly on odd numbers hence we consider like a Collatz sequence consists of only odd numbers.

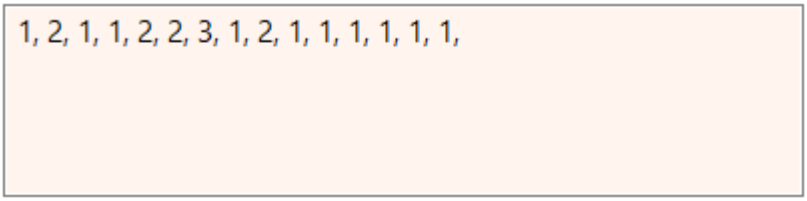
The first line is the ***odd number*** *N* itself and the bottom line is 1. The numbers in parenthesis are branch orders (ordinal numbers). For example 5 is at the 1st branch of 1 and 13 is at the 2nd branch of 5⋯ 16385 is at the 1st branch of 12289.

Get the number

The input of **Get the number** is ***branch order sequence*** entered in **I/O frame** and it outputs ***Reversed Collatz number sequence*** into **Output frame** and the terminal node number of the branch order sequence into ***odd number***.

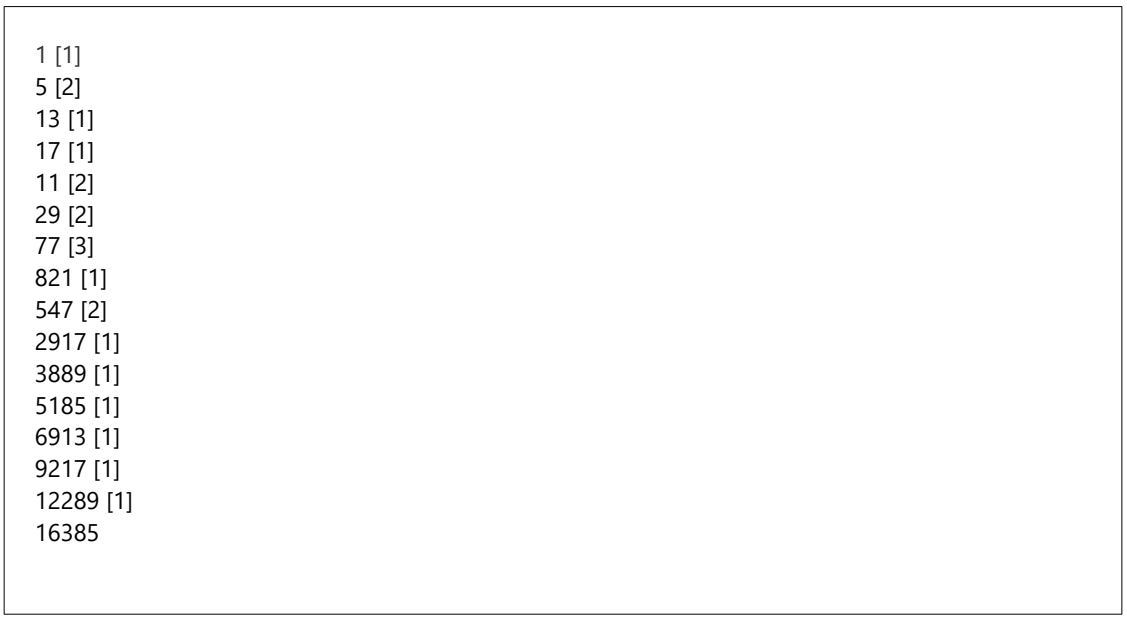


You can enter an arbitrary sequence of natural numbers separated comma with any length (theoretically but actually the maximum text length is limited to 32767) in **I/O frame** as ***branch order sequence*** like below. The sequence is always assumed to start at root node 1 in the Collatz Tree.



The Collatz Tree is an infinite graph and every node on the tree has infinite number of branches except leaves of the tree. Hence a branch order sequence is essentially an arbitrary sequence of any numbers but it might not be valid entirely because the sequence might halt at a multiple of 3. In such a case it eliminates an invalid part of the sequence and leaves a shortened sequence that terminates at the multiple of 3.

Get the sequence and **Get the number** are absolutely inverse functions. The former outputs a ***branch order sequence*** from an ***odd number*** and the latter outputs an ***odd number*** from a ***branch order sequence***. Indeed Reversed Collatz number sequence is looked like the upside down of the Collatz number sequence.

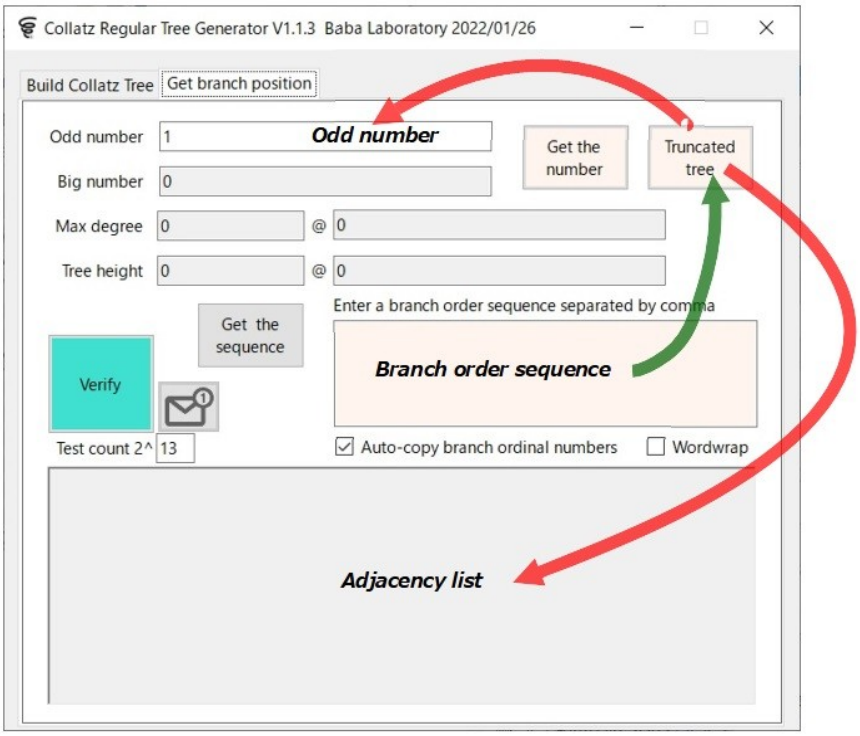


The numbers in parenthesis are branch orders (ordinal numbers). If the checkbox **Export Zelkova Tree CSV** is ON in **tab A** then the result is saved in "***CollatzNumber.csv***" file on the desktop of your PC in Zelkova tree CSV file format.

Truncated tree

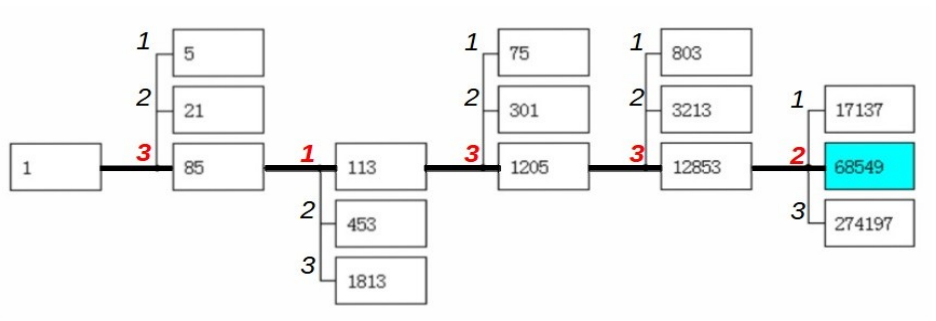
The input of **Truncated tree** is a ***branch order sequence*** entered in **I/O frame** and it outputs an ***adjacency list*** of the truncated tree in **Output frame** and the terminal node number *N* of the branch order sequence in ***odd number*** box. The ***branch order sequence*** is assumed to start at root node 1 of the Collatz Tree. A truncated tree stands for a subtree of a Collatz Tree such as a chain of nodes plus leaves sticking to

the stem directly. The stem is an elementary path from node 1 to terminal node *N*.



Given a ***branch order sequence***, output the terminal node number of the sequence into ***odd number*** and an adjacency list of the truncated tree into **Output frame**. If the checkbox **Export Zelkova Tree CSV** is ON in **tab A** then the result is saved in "***TruncatedTree.csv***" file on the desktop of your PC in Zelkova tree CSV file format.

In general a regular Collatz Tree is too large to manipulate but a truncated tree is much more compact and makes handling easier. Furthermore the truncated tree gives you an insight what is a ***branch order sequence*** and why it is so important. Look at the chart below. This is the truncated tree for odd number 68549.



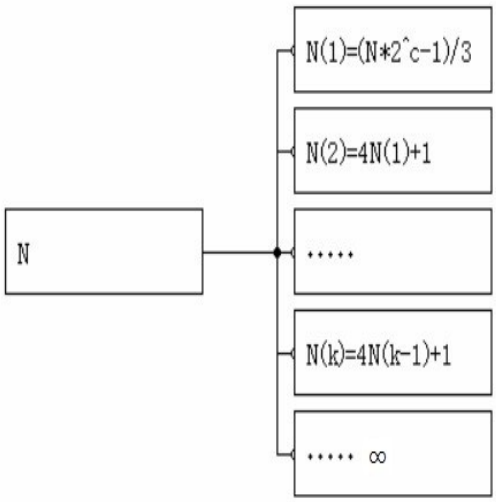
It is so clear that the branch order sequence of 68549 is (3, 1, 3, 3, 2) as you read on the stem line from 1 to 68549.

A ***branch order sequence*** is a sort of universal address for any node in the Collatz

Tree. You can send a mail to any odd number *N* in the infinite universe just putting this mailing address line on the envelope. It might be an ultimate my-number system.

Collatz Tree Structure

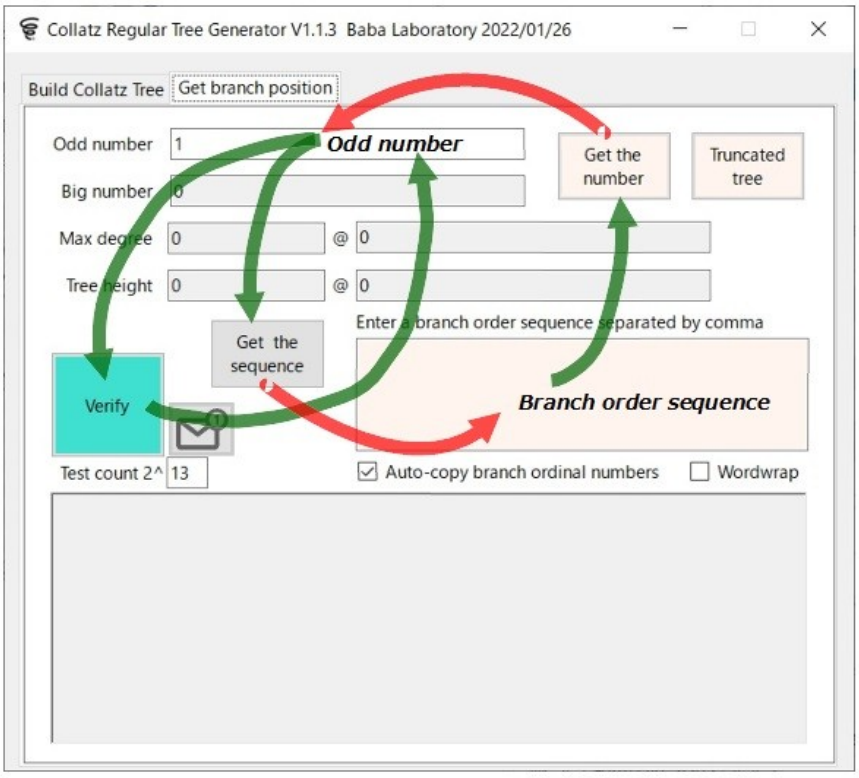
How to get this address line or how to decide the Collatz Tree structure is the top secret of the Conjecture. We discovered it through one month talk marathon at an online forum around scientific topics. Yes, it was not a breakthrough invention but something like picking up a lost.



As you see the Collatz Tree structure is quite simple and definitive. Every node except a multiple of 3 in the Collatz Tree shares this structure. You can build the whole Collatz Tree by just starting from *N*=1. In fact the Collatz Tree Generator is using this function recursively and building the whole body of a regular Collatz Tree in a deterministic way.

Verification Test

Verification Test verifies the Collatz Tree structure on which the Collatz Tree Generator is entirely based by testing every odd number in a designated range. The experiment starts at ***odd number*** and repeats the test in ***test count*** times.

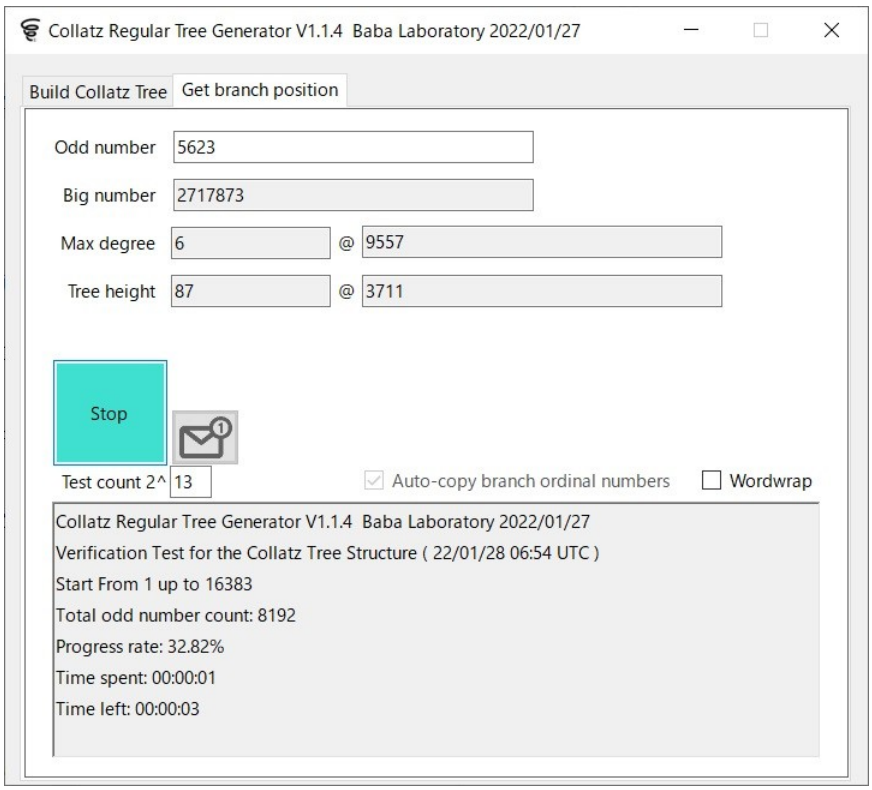


Actually the test is performed by the combination of 2 functions, **Get the sequence** and **Get the number**. It works like:

odd number N → **Get the sequence** → *branch order sequence* → **Get the number** → *odd number N'*

Verification Test verifies whether the input number *N* and the output number *N'* are equal or not then increment the ***odd number*** by 2 if the result is OK otherwise it will show an alert message and halt.

While the experiment is running the layout of the display window is significantly changed. ①Other functions are disabled by hiding three buttons, **Get the sequence**, **Get the number** and **Truncated tree**, ②**I/O frame** is hidden to avoid the interference during the experiment, ③**Auto-copy branch ordinal numbers** check box is frozen to keep it checked.



During the experiment the progress information is updated in **Output frame**. It includes *current version and release date of the program, starting date and time in UTC, the start number and the last number, total odd number count, progress rate, time spent and time left.*

After the experiment **Output frame** turns to appear like below.

Collatz Regular Tree Generator V1.1.4 Baba Laboratory 2022/01/27
Verification Test for the Collatz Tree Structure (22/01/28 06:49 UTC)
Start From 1 up to 16383
Total odd number count: 8192
Big number : 9038141
Max degree: 8 @ 120149
Tree height: 101 @ 13255
Progress rate: 100.00%
Time spent: 00:00:05
Verification Test Complete !

Click the mail button above and send the result to us (babalabos@outlook.jp) !!
If you have any additional information, use the colored textbox as a sticky note.
Thanks in advance...

Regardless whether the experiment succeeded or not, the mail button becomes active and you can send the result to the author. The body of the mail includes ①contents in **Output frame** and ②contents in **I/O frame**. Any other information is never transferred to the outside. You can add any additional information in **I/O frame**. This button is active only immediately after a **Verification Test** and can be pressed just once. Thank you for your help in advance.

Credit

Thanks for everyone who have helped me during this critical period of my steeply declining life time. I really owe to the young mathematician, my eternal friend Noureddine Mohiti in Morocco who inspired me like a flash of lightning in the darkness. Thomas Edison said “Genius is 1 percent inspiration and 99 percent perspiration”. Of course I’m not a genius and have never had this result without his sparkling 1% inspiration as well.

Appendix I

Proposition: As far as the Collatz graph is connected (as an undirected graph) it does neither circulate nor diverge other than cycle(1, 4, 2).

Proof: Let $G(V, E)$ be the Collatz graph such as V is the set $\mathbb{N}$ of natural numbers and E is a set of directed edge $e(i, j)$ where if i is even then $j = \frac{i}{2}$ otherwise $j = 3i+1$. Apparently every vertex $v(i)$ in G has only one outgoing edge $e(i, j)$.

Assume that G is connected and it has two cycles c_1 and c_2. Since G is

connected there must be at least one undirected path $p(v_1, v_2, \cdots, v_i, \cdots, v_j, v_k)$ between c_1 and c_2 while v_1 is a vertex on c_1 and v_k is a vertex on c_2. Obviously the orientation of $e(v_1, v_2)$ and $e(v_j, v_k)$ are opposite. Hence there must be a vertex v_i which has two outgoing edges. This contradicts to the premise.

Similarly infinite divergence case is also rejectable. Therefore since the graph *G* already has a cycle (1, 4, 2) it does neither have any more cycles nor diverge into an infinite space. Thus Collatz conjecture is true as far as the Collatz graph is connected. QED

Appendix II

Proposition: Let odd number *n* be a multiple of 3. Then there is no trajectory (a sequence of Collatz mapping) from any other odd numbers to the number *n*.

Proof: Let odd number $n = 3m$ where *m* is an arbitrary odd number. From the definition of Collatz graph we have an infinite even number sequence $S = (2n, 4n, \cdots, 2^c n, \cdots \infty)$ connected to the node *n*. Assume that the proposition is false then there must be an even number *Y* in the number sequence *S* such as $Y = 2^c n$ and $Y = 3X+1$ where *X* is an odd number. Then we have an equation:

$2^c n = 2^c \cdot 3m = 3X+1$

$$m = \frac{X+\frac{1}{3}}{2^c}$$

Apparently there is no such integer *m*. Thus the assumption does not hold. In other words a multiple of 3 is a leaf of Collatz Tree. QED

2022/01/26 Fukaya city, Japan
Michiro Nasu a.k.a. Baba Age
Baba Loaboratory Inc.
Zelkova Tree Tent Village
babalabos@outlook.jp