Math
Darrin Taylor
July 8, 2019

Collatz 3n+1 conjecture

A proof that 1->4->2->1 is only possible loop.

Initial definitions needed to proceed

The Collatz conjecture states

Take any integer A0

If even divide by 2

If odd multiply by 3 and add 1

And it will always decay into the loop 1->4->2->1

There are two necessary steps to prove the conjecture:

1) Prove that no other loops are possible
2) Prove that once no other loops are possible that the sequence cannot infinitely ascend.

The approach here is to apply base 3 mathematics on partially known numbers to prove the first step. So the math side of it is a little different and it reads a bit like a logic puzzle from computer science instead of normal mathematics.

First lets define a starting number:

A0

They key to this approach is to take A0 and convert it into base 3 number.

Base 3 is like binary but instead of 0 and 1 it has 0,1,2 and each digit is 3 times the next lower digit.

First consider a number

We will use A0=308 base 10 to make a point about forward and backwards.

308 base 10 is expressed as 102102 base 3

This number is even so we divide to
A1=154 base 10 expressed as 012201 base 3

The forward direction is from A0->A1 and divides by 2; the backwards direction is from A1->A0 and multiplies by 2. Some things are easier to see in the backwards direction other things are easier to see in the forwards direction. This proof will use the words forward and backwards to mean forwards in time and backwards in time using the known rules of divide by 2 and 3n+1 in the forward direction and multiply by 2 and (n-1)/3 in the backwards direction.

We have chosen base 3 because it vastly simplifies the 3n+1 and inverse (n-1)/3 that are used moving forwards or backwards through a collatz sequence or collatz loop.

SHIFTS DO NOT CHANGE ANY DIGIT VALUE EXCEPT TO SHIFT THEM EXCEPT LEADING ZERO AND LEAST SIGNIFICANT DIGIT.

Simple Rules of base 3:

1) Even or odd is determined by an even number of 1s as both 0 and 2 are even.
2) Multiply by 3 is a shift to the left.
3) 3n+1 is a shift to the left and the least significant bit becomes 1
4) Divide by 3 is a shift to the right.
5) (n-1)/3 is a shift to the right that destroys the least significant bit (which MUST always be a 1 because we are just doing the reverse of step 3 above)

Understanding base 3 digit length

There are only 2 operations that change the total number of base 3 digits in the forward direction.

1) 3n+1 shifts the digits up 1 and adds a 1 in the least significant digit and this increases the total number of digits by +1
2) Dividing a number that has a leading 1 decreases the total digits by -1

The important aspect to realize here is that if you count the number of divides of numbers that start with leading 1s and you count around the entire loop you will have the correct total number of shifts around the whole loop. This is because the loop must return to the starting integer and to the starting integer's total digit length. So the change must be 0 so total shifts must equal total leading 1s divided.

Because the shifts 3n+1 or (n-1)/3 are difficult to predict it is sometimes easier to never write them down and recover the information from counting the leading 1s.

So a technique exists where we move backwards in time and just have a sequence of multiply by 2 and know the number of shifts by virtue of counting the number of leading 1s then we can say things about the loop ONLY from the leading digits for each number in the sequence.

Although there is no clear pattern regarding when shifts occur, there is a VERY clear pattern regarding when leading 1s occur and how often digits can be lost moving forward from dividing leading 1s.

Moving backwards the same logic can apply provided the numbers we are dealing with are large enough that the total digits used stays below a total number magnitude of the 2^60s that have already been numerically eliminated in previously published attempts at this problem. We will need less than 30 base 3 digits so this will not be a problem.

Pre-View of Proof

Proof that there are no loops except 1 -> 4 -> 2 -> (1) base 10

1) Moving Backwards using base 3 consider the most significant digit 1# or 2# and prove that there are only 2 patterns possible that when converted to their Forwards form are:

Segment type A

1# -> 2# -> (1#)

The (1#) is in the next segment shown because of the "->" multiply by 2 after 2#.

There is one leading 1 so there are one non local "shifts" created by this segment.

This represents 2 divides (->) and 1 shift (3n+1 or (n-1)/3)

and

Segment type B

1# -> 2# -> #1 -> (1#)

The (1#) is the next segment shown to show 3 divides (->) in-between numbers

There are two leading 1 so there are two non local "shifts" created by this segment.

This represents 3 divides (->) and 2 shift (3n+1)

2) Show that unless Segment B can occur 3 times in a row that the loop can only descend except for very small numbers smaller than the search space already explored via numerical methods. In other words, show that for N larger than 1000 that Segment B must be permitted to occur 3 times in a row or the loop will descend. ABB is 8 divides and 5 shifts and descends.

3) Because Segment A descends and B ascends and ABBABB….ABB descends we must have at least one segment of BBB in a row. Looking at the point where this segment mates with a previous descending segment there must be a segment ??ABBB otherwise BBB ascends and never returns to the starting point. We will first prove that AABBB is impossible. Then we will prove that ABABBB is impossible. This leaves us with ABBABBB which is also shown to be impossible. Now we have an inductive proof that for a given BBB there is always a sequence BBBABBB and (ABBB) ascends. This means that as we cycle around the loop we can only "pass and expell" segments that ascend and can never "pass and expell" a segment that descends.

BBB "expells" (ABBB) then "expells" (B) segments until it reaches an A segment and then we find that per our proof above of ABBB implies BBBABBB we again have another (ABBB) segment to expel and are again looking for an A and expelling individual (B) segments until we find one only to expel (ABBB) again. As we cycle around the entire loop we cannot find a segment that is expelled that descends. Thus the loop infinitely ascends.

Argument that infinite ascent is unlikely (starts on page ~70)

1) Show for non trivial sequences an ending of #2 or #01 always occurs

2) Show that once an ending of #2 or #01 is reached that there are only 16 segments of possible branching and all return to #2 or #01.

3) Show that 15 of the 16 pathways descend and characterize the only one that doesn't descend.

4) Show that in Base 4 the average expected value before a 3n+1 to the next 3n+1 is negative and that although the rising segment can repeat the falling segments can also repeat and the pattern revolves so that each ending of base 4 double digits will tend to be cycled through without an obvious bias to overcome the ~factor of 4-5 bias toward descent.

Base 3 Math Tricks

Because the shift operations do not change any digits except the leading 0 and least significant digit it allows us to do math even if we only have partial knowledge regarding the number we are doing math on.

Now we will look at the multiply and divide tables in base 3 and review a few tricks that we will use to apply math on partially known numbers.

Dividing by 2	**In base 3**			
Line	Value of remainder from Digit one higher carries in and adds 3	Digit in location A at T=0	Digit in location A at T=1	Remainder
1	0	0	0	0
2	0	1	0	1
3	0	2	1	0
4	3	0	1	1
5	3	1	2	0
6	3	2	2	1

Multiplying by 2	**In Base 3**			
Line	Value if remainder from Digit one lower carries in and adds 3 which is a 1 to this digit	Digit in location A at T=1	Digit in location A at T=0	Carry up to next significant digit
1	0	0	0	0
2	0	1	2	0
3	0	2	1	3
4	1	0	1	0
5	1	1	0	3
6	1	2	2	3

There are several take aways that are very important in these tables.

Refer to the table above and the picture on page 8 to help follow my reasoning.

1) In the forward direction because we are only dividing when there are an even number of 1s there will always be an integer number of pairs of 1s. The first 1 in the pair initiates what I call a carry zone, all the digits inside that zone have a remainder carried down to them providing them with +3. The 2nd 1 in the pair ends this "carry zone" (until the next pair of 1s) as such we can think about the pair as initially 1…1 then converted to (—>) 0…2. I call the pairs of 1,1 "border" because it is the border between the "carry zone" and the "not carry zone". The border is Line 2, Line 5 in the Divide table.

2) Once inside a "Carry zone" there is no digit that can produce a 0 in its digit in the next iteration. This is the primary tool used later to disprove ascent segments are possible. Carry zone is defined as occurring after Line 2 in the divide by chart and it includes Line 4,6 and the Border at Line 5.

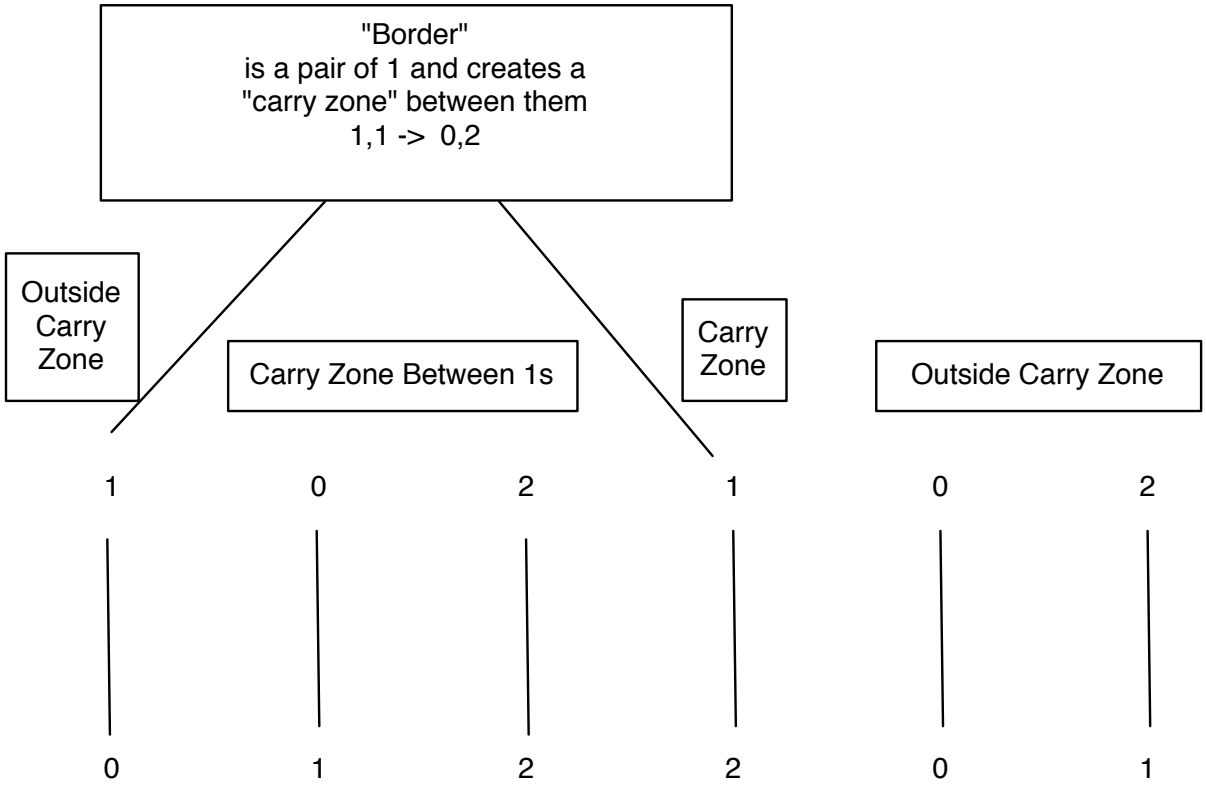
3) Likewise when outside a "Carry zone" there is no way for the future digit to be a 2 in next iteration without a 1 first changing back to a carry zone.

4) Lastly it is very important to note that in the forward direction the only way to reduce the total number of digits is to have a leading 1 and divide it by 2. This balances with the 3n+1 operation which is the only way to increase the total digits. The 3n+1 "Shift" adds to total digits by +1 digit and divide of leading 1 reduces the total digits by -1. This is important because in a loop there must be a 1 to 1 correlation between the number of 3n+1 shifts and the number of leading 1s during a divide operation. Moving backwards a new way of evaluating is to just multiply by 2x or 2x +1 due to carried up from 2X below the known numbers in question. It is possible to just evaluate successive multiple by 2x or 2x+1 and ignore the shifts and rely on recovering the number of shifts from the leading 1s. This technique appears to be novel to this paper (as it allows proving loops are impossible).

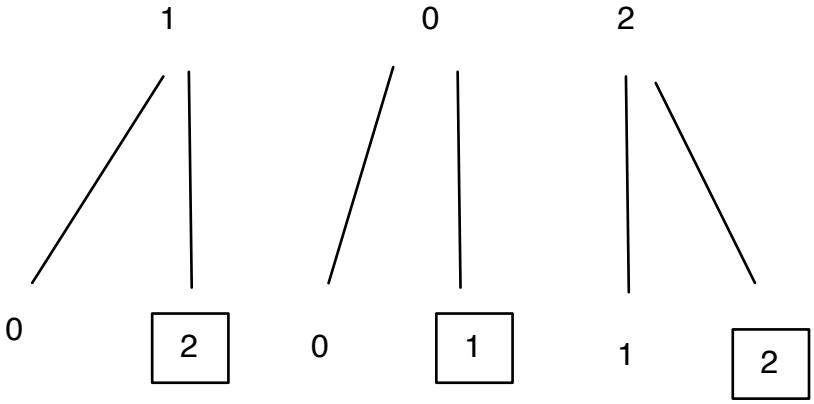
Base 3 relationships

Graphically showing base 3 divide by 2 and multiply by 2 relationships

Here is a graph showing top before and bottom afterwards. 1 divided by 2 is 0 remainder 1 which starts a carry zone by carrying the remainder down. (0+3)/2=1 and carried down. (2+3)/2=2 and carries down. 1+3 divided by 2 is 2 but ends a carry zone. 0 divided by 2 is 0. 2 divided by 2 is 1.

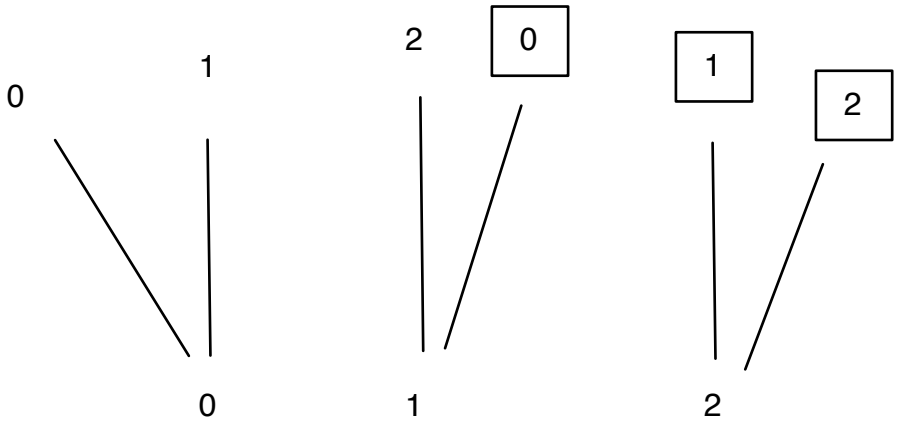


Moving in Forward Direction
If past the 1st 1 in a pair we are in a carry zone
If in a carry zone the output is in square



Notice there are no 0s as possible
output when inside a carry zone
And no 2s when starting outside a carry zone
are possible as output

Moving in Backwards Direction (bottom to top)
Options are x2 and x2+1
If output is in square top row must be in a carry zone



Notice 2s always go backwards to inside a carry zone or the 1 that exits a carry zone.

Notice that 0s must go backwards to outside a carry zone or the first 1 of the border

Some patterns when dividing Base 3

by 2

Digits containing even number of 1s	Digit IS non-carry zone
	Below Can never be 2

		Examples		
Digits containing even number of 1s	Digit IS non-carry zone	1	2	0
Digits containing even number of 1s	Digit IS non-carry zone	0	1	0
	Below Will always be 0	Below Will always be 0	Below Will always be 0	Below Will always be 0

Digits containing odd number of 1s	Digit IS in a carry zone
	Below Can never be 0

		Examples		
Digits containing odd number of 1s	Digit IS in a carry zone	1	2	0
Digits containing odd number of 1s	Digit IS in a carry zone	2	2	1
	Below Will always be 2	Below Will always be 2	Below Will always be 2	Below Will always be2

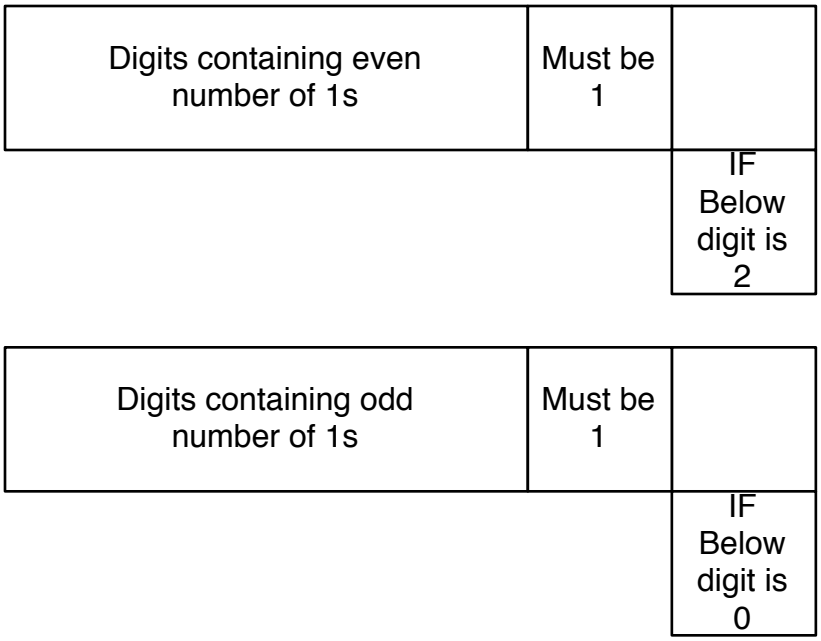
Some Base 3 patterns when dividing by 2

		Counter Examples	
Digits containing even number of 1s	Must be 2	1	0
Digits containing odd number of 1s	Must be 1	0	0
	IF Below digit is 2	Can only be 1	Can only be 1

		Counter Examples	
Digits containing odd number of 1s	Must be 0	1	2
Digits containing even number of 1s	Must be 1	2	2
	IF Below digit is 0	Can only be 1	Can only be 1

Blocks to avoid having illegal values below Carry or Non Carry zones.

1s are used to either start a carry zone or end a carry zone to prevent a carry zone above a 0 or a non carry zone above a 2 which are violations and indicate a disproven case.



The known spaces below can tell us if we are in a carry zone even if we have empty digits above.

Note that we can see a violation occurred because the non carry is above a 2 before it can be above a 0 needed to exit the non carry state.

Digits containing even number of 1s	Cannot Exit Non Carry	Cannot Exit Non Carry	IF 1 This Exits
	1	2	0

Digits containing odd number of 1s	Cannot Exit Non Carry	Cannot Exit Non Carry	IF 1 This Exits
	1	0	2

Likewise the Carry zone cannot exit before crossing above the 0 causing a violation because to exit requires a 2 below a 1.

Repeated 0s or 2s force the line above to have 1 of the 0s or 2s as follows:

In a non carry zone either a 1 or a 0 can create a 0 below as 1 divided by 2 is 0 and 0 divided by 2 is also 0. But the 1 creates a carry zone and 0 under a carry zone is a violation and must be avoided so we know the left most 0 was caused by a 0 above it.

Digits containing even number of 1s	Must be 0	Could be 1 or 0
	If 0	And 0

IF 1 above first 0 it would make carry zone that is violated by being above 2nd 0

Digits containing odd number of 1s	Must be 2	Could be 1 or 2
	If 2	And 2

IF 1 above first 2 it would make non carry zone that is violated by being above 2nd 2

In a carry zone either a 1 or a 2 can create a 2 below as 1+3 divided by 2 is 3 and 2+3 divided by 2 is also 2 (and carry). But the 1 creates a non carry zone and 2 under a non carry zone is a violation and must be avoided so we know the left most 2 was caused by a 2 above it.

Non Localized Interactions based on where 1s can be

The violation of a 2 under a non carry zone can only be avoided if the non carry zone can be converted into a carry zone before reaching the 2. But the first 1 requires a 0 underneath it. Sometimes we know the middle square is a 1 or a 2 but cannot be a 0 and thus can infer that the 0 must have a 1 above it.

Digits containing even number of 1s	So This must be 1	??	This would Violate
	0	1	2

Digits containing odd number of 1s	So This must be 1	??	This would Violate
	2	1	0

Same logic except violation caused by 0 and 2 needed under 1.

Consider 1# in Reverse

We will start by considering a base 3 number with a start digit of 1 and unknown digits and unknown digit length denoted as #. Because numerical methods have been used to show that all numbers smaller than 2^60 return to 1 we will not worry about running out of digits and emptying #. We will only need 20 or 30 digits to make our case.

Moving backwards in time divide by 2 becomes multiply by 2. In base 3 the 3n+1 shift just shifts up the digits adds a 1 to the unknown number's least significant digit. But this doesn't change the order or content of the "known" digits. So we will only multiply by 2 and use the leading digit to infer how many shifts have occurred and will completely forget about the (n-1)/3 operation as it can be recovered by using the leading 1.

This is possible because a Collatz loop sequence of numbers in base 3 has a starting position that returns to itself and therefore a loop must have increased digits (shifts) that are offset by decreased digits via leading 1s divides. Shifts are the only thing that increases digits and it is always by a single digit in base 3. Divide by 2 of a leading 1 is the only thing that decreases digits and it is also always a single digit. Because we are only considering multiplies and leaving all the shifts un-shown there is therefore a 1 to 1 correlation between a leading 1 occurring and a shift occurring somewhere in a loop. This logic ONLY works for sequences that start and end with the same number of digits. We will need different logic when we talk about non loops or segments that change total number of digits.

Because all of the pathways will eventually descend we will use a larger substitution for 3n+1. Instead of calculating the value we will substitute a multiple that is always larger for numbers above 1000. We will use 3.001n which is equal to 3n+1 for n=1000 and larger than 3n +1 for all n>1000. n<1000 have already been numerically proved to descend to 1 and then be part of the only known loop. Because 3.001n is multiplied and divide by 2 is multiplied by 0.5 we can just rearrange so that the substituted 3.001n is always accounted in the segment that predicted it. This allows a shift value of the segment that is larger than 3n+1 in a loop to be considered and this segment is still shown to always descend.

The backwards multiplication tree in base 3.

Start with 1# and multiply it by 2 to find the previous first digit.

There are two possibilities:

2*1# and 2*1# + carry up from unknown number.

****

We pause for a moment to consider the least significant 2 digits which we will multiply by 2 and see if they carry up more than 1. I will show the numbers as numbers in base 3 but larger than 2 so that you can easily see what the value is and pay careful attention to the carry process which must bring each digit down to 2 or smaller by carrying up a full next most significant digit which is 3 base 10 or 10 base 3.

#22 *2 = #44

carry up the least significant 4 by subtracting 3 to get 1 and add 1 to the next digit.

#22 *2 = #51

the 5 is smaller than 6 so only a single carry up exits in the numbers we can see

#22 *2 = #21 + single carry up of #100 added to the unknown 3rd least significant digit.

*****

Returning to:

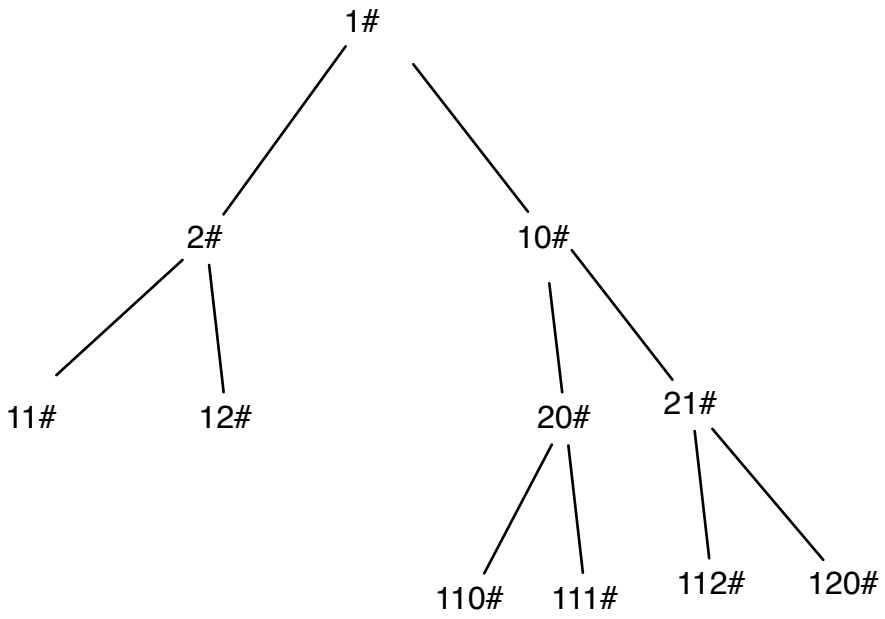
2*1# and 2*1# + single (1) carry up from unknown number.

we get 2 possible values in base 3

2# and 10#

Here 1#*2+1#=3# and 3# in base 3 is 10# and the unknown part of the number remains so 10#.

This Tree shows the possible values in base 3.



Notice that all the numbers on the bottom line follow the pattern of the top 1# if we just forget their values except for the first digit and start over at the top.

There are only 2 types of backwards segments in a loop if we sort by most significant digits.

1#->2#->(1#) from the left side of the tree

and

1#->1#->2#->(1#) from the right side of the tree

Here the (1#) represents the first value of the next segment.

-> represents a multiply operation or a divide operation if moving forward

Leading 1s represent a shift that must occur to balance out the loss of digit in the forward direction.

So loops are built in the backward direction by either

1 shift to 2 Divides which will always descend if we consider the forward direction.

or

2 shifts to 3 Divides which always ascends if we consider the forward direction.

If you wonder what happens if you start with 2#.

If you start with 2# you can multiply to get 11# or 12#. Then you simply reposition your point of view and start with the 1# using the diagram above. The leading digit is always either a 1 or a 2 as a leading 0 can be ignored.

Forwards vs Backwards

The tree generated is moving backwards but we are interested in the tree moving forwards. We can get the forward loop by starting at the bottom of the tree and moving towards the top. So we must reverse the segments to get:

1#->2#->1#->(1#)

and

1#->2#->(1#)

Minimum Forward ascending segments.

Lets name Segment A as:

1#->2#->(1#)

Notice 2 divides and 1 shift caused by 1 leading 1# because (1#) is in the next segment.

Lets name Segment B as:

1#->2#->1#->(1#)

Notice 3 divides and 2 shifts caused by 2 leading 1# because (1#) is in the next segment.

Now consider the sequence ABB..ABB…ABB is the most ascending sequence we can create without putting 3 Bs in a row. Order doesn't matter ABB…ABB or BAB…BAB or BBA…BBA they all create the same result 2 Bs in between every pair of As.

The ratio for ABB is

3 Divides and 2 Shifts From Segment B

plus

3 Divides and 2 Shifts From Segment B

plus

2 Divides and 1 Shifts From Segment A

Which is

8 Divides and 5 Shifts

This gives

divide by 2^8= 256 and multiply by 3.001^5 = 243.405 and descends in the forward direction.

For n above 1000, Any loop that does not contain at least one stretch of segments of BBB must descend. For N=1 base 3 [1]->11->2->[1]->(11) only needs Segment A this is the 1->4->2->1 segment in base 3 with is made exclusively of segment A.

Any loop that contains only Segment Bs ascends and cannot loop. Substituting 3n for 3n+1 so that the substituted number is always smaller than 3n+1 we have

3 Divides = divide by 8 and 2 multiply by 3 for multiply by 9.

Despite using a multiple always smaller than 3n+1 the sequence always ascends.

So we cannot have a loop made from segment B, or segment A and BBA descends.

Adding an extra B to ABB…ABB… must produce at least one BBB

Later we will consider ABBB

The ratio for ABBB is

3 Divides and 2 Shifts From Segment B

plus

3 Divides and 2 Shifts From Segment B

plus

3 Divides and 2 Shifts From Segment B

plus

2 Divides and 1 Shifts From Segment A

This is 11 Divides and 7 Shifts.

Using 3n instead of 3n+1 which is smaller this sequence still ascends.

2^11=2048

3^7=2187

The shifts increase the value more than the divides decrease it and this segment ascends.

Method of disproving specific sequences

Below we will take long sequences that are identified only by the first digit for each step. This will be shown in the forward direction from top to bottom. All shifts will be hidden from view inside the # which starts off to the right of the tables shown. The reason for this is that when a sequence is divided it is MUCH easier and more natural to keep the digits in the correct relative positions

121#######
022#######

VS

121#######
022#######*
*=shift

In the first case you can clearly see that the leading 1 became a 0 and the new leading digit is the 2 to the right accordingly. The lines of causation are straight up and down.

In the second case the same lines of causation connect the same digits but the shift that occurred has made the lines of causation diagonal and it is much harder to understand what is going on.

As a more practical consideration we do not know where the shifts are and could not place them because of this. Leading 1s do predict how many shifts occur but they do not predict where the shifts occur. So please find the digits arranged in the tables below so that the lines of causation run straight up and down. Also note that to do this digits need to proceed from top left to bottom right as divides decrease the digit count.

Numeric Method of disproving Loop

The method of disproving the theoretical loop below will be to start with the BBB case and move this BBB around the loop. By that I mean we will consider the sequence before the BBB and if that sequence is a B giving us BBBB we will "expell" the right most B and look for another sequence on the left side which may be an A or a B.

As we proceed we will expel only segments of (B) or later segments of (ABBB). Both of these segments ascend the in forward direction. We will find it is impossible for us to "expel" or pass any sequence except these sequences. Because we are looking at left most digits and moving to the left when we "expel" ascending segments we will eventually rotate around the entire loop having only "expelled" ascending segments. Thus the loop must ascend.

Starting with BBB we will proceed by BBBB expelling (B) and shifting until we have ABBB. When we find ABBB we know that it is either AABBB or BABBB. The first proof will be that AABBB is impossible. This will be done by looking at ABBB with the assumption that the digit just before the first A must be a 2 as needed for an A and not a 1 as needed for a B. This restriction will make the sequence impossible via forcing mathematical violations which render the position illegal. This same technique will be used for each subsequent disproof.

Starting with BABBB we must have either have BBABBB or ABABBB. The second proof will be that ABABBB is impossible. Again by using BABBB and forcing the digit before the first B to be 2 for A and finding this is illegal.

Starting with BBABBB we must either have ABBABBB or BBBABBB and the final proof will be that ABBABBB is impossible. Again this will be done using BBABBB and a single digit before the first B that will be forced to be 2 for A and this will be shown to be illegal.

First prove AABBB is impossible in forward direction

The values in column 1 to 7 are knowable unknown digits. We can apply the tricks we have learned to derive these digits.

Note that the leading digits are X(1#, 2#)(1#, 2#, 1#)(1#, 2#, 1#)(1#, 2#, 1#)1#

X is not shown and would be (#1,#2) for A or (1#, 2#, 1#) for B. But we can instead use the last digit so column/row 0/0=2# for A and 0/1=1# for B.

I have highlighted the squares changed and will provide detailed explanation for each discovered digit. We will read the table and then the description showing how the highlighted digits were discovered.

Column	0	1	2	3	4	5	6	7	8
Row 0									
Row 1		1							
Row 2			2						
Row 3			1						
Row 4				2					
Row 5				1	0				
Row 6					1				
Row 7						2			
Row 8						1	0		
Row 9							1		
Row 10								2	
Row 11								1	0
Row 12									1

Notice that the pattern ABBB, 1->2->1->2->1->1->2->1->1->2->1->(1) is shown above.

In Column / Row positions 1/1, 2/2, 2/3, 3/4, 3/5, 4/6, 5/7, 5/8, 6/9, 7/10, 7/11, 8/12.

Also notice that regardless if the next segment is A or B the digit in 8/12 must be a 1

The leading 1s decrease the digits which require shifts to bring them back to the starting point but we are not considering shifts right now. **No shifts are shown.**

Location Column / Row

Location 4/5, 6/8, 8/11 must all be 0 because they are inside a carry zone created by the 1 at 3/5, 5/8, 7/11 and the only way to make the next iteration digit a 1 inside a carry zone is a 0. This is because 0+3 divided by 2 is 1 with a remainder / carry down.

Column	0	1	2	3	4	5	6	7	8
Row 0									
Row 1		1							
Row 2			2	2					
Row 3			1	1					
Row 4				2					
Row 5				1	0				
Row 6					1				
Row 7						2			
Row 8						1	0		
Row 9							1		
Row 10								2	
Row 11								1	0
Row 12									1

Location Column / Row

3/3=1 because a carry zone caused by the 1 at 2/3, under a non carry zone caused by no 1s on row 2, over a 2 must always be 1. The non carry zone can only place 0 or 1 in 3/3 and the carry zone can only turn the 1 into a 2.

3/2=2 because a non carry zone above 1 is always 2.

Column	0	1	2	3	4	5	6	7	8

Column	0	1	2	3	4	5	6	7	8
Row 0									
Row 1		1	2						
Row 2			2	2					
Row 3			1	1					
Row 4				2					
Row 5				1					
Row 6					1				
Row 7						2			
Row 8						1			
Row 9							1		
Row 10								2	
Row 11								1	
Row 12									1

Location Column / Row

2/1=2 this is because 2/2=2 and 3/2=2 and although 2/1 could be 1 or a 2 and create a 2 at 2/2=2 this would end the carry zone and the non carry zone at 3/1 would be above a 2 and this is a violation. Without the +3 carry from the carry zone even the largest digit 2 is divided down to at most 1 and cannot provide the 2 needed in 3/2=2. But inside a carry zone 1+3 divided by 2 provides a 2 as does 2+3 divided by 2. We remain unsure of the contents of 3/2.

Column	0	1	2	3	4	5	6	7	8
Row 0	1	0	1 or 2	?	?	?	?	?	?
Row 1		1	2						
Row 2			2						
Row 3			1						
Row 4				2					
Row 5				1					
Row 6					1				
Row 7						2			
Row 8						1			
Row 9							1		
Row 10								2	
Row 11								1	
Row 12									1

Location Column / Row

I have added a column 0 because we have enough information to disprove AABBB.

We are going to move BACKWARDS from row 1 to row 0. This involves either multiplying by 2 or multiplying by 2 and carrying up an additional 1 from the unknown portion of the number in row 1.

In this case, we will use the smaller number because we are proving our number is too large. The first two digits of row 2 are 1*3 +2*1=5 in base 10. Multiplying by 2 we know that the first 3 digits of row 0 starting at 0/0, 1/0, 2/0 must hold 10 or 11 base 10. This gives in base 3 101# or 102#.

0/0 being a 1 means that the segment we created ?ABBB must be BABBB which means that AABBB is impossible. Note that the seed segment was (12)(121)(121)(121)1 which is ABBB with the next 1 which is common to whatever is after the 3rd B. But the addition of the 1 at 0/0 means that the extended segment must be [121](12)(121)(121)(121)[1??] and NOT [12](12)(121)(121)(121)[1??]. QED AABBB is impossible and must be BABBB.

First prove ABABBB is impossible in forward direction

This segment is ONLY considering the possibility of ABABBB segments existing. We will disprove this by showing that given BABBB we must have BBABBB.

Column	0	1	2	3	4	5	6	7	8	9	10	11
0												
1		1										
2			2									
3			1	0								
4				1								
5					2							
6					1							
7						2						
8						1	0					
9							1					
10								2				
11								1	0			
12									1			
13										2		
14										1	0	
15											1	
16												
17												
18												
19												
20												

Location Column / Row

3/3=0, 6/8=0, 8/11=0, 10/14=0 because Carry zones above a 1 is always a 0

Colum 0 1 2 3 4 5 6 7 8 9 10 11

Column	0	1	2	3	4	5	6	7	8	9	10	11
Row 0												
Row 1		1	1									
Row 2			2									
Row 3			1	0								
Row 4				1								
Row 5					2							
Row 6					1							
Row 7						2						
Row 8						1	0					
Row 9							1					
10								2				
11								1	0			
12									1			
13										2		
14										1	0	
15											1	
16												
17												
18												
19												
20												

Location Column / Row

2/1=1 ABABBB is disproven if row A is larger than 111111111…# .

We need row 1 to be small enough that when moving BACKWARDS from row 1 to row 0 and multiplying by 2 we do not force a carry into 0/0 making it 1 which would make the leading segment B and make the scenario BBABBB which proves ABABBB illegal.

Colum	0	1	2	3	4	5	6	7	8	9	10	11

Column	0	1	2	3	4	5	6	7	8	9	10	11
Row 0												
Row 1		1	1									
Row 2			2									
Row 3			1	0								
Row 4				1								
Row 5					2	2						
Row 6					1	1						
Row 7						2						
Row 8						1	0					
Row 9							1					
10								2				
11								1	0			
12									1			
13										2		
14										1	0	
15											1	
16												
17												
18												
19												
20												

Location Column / Row

5/6=1 because 5/5 limits it to 0 or 1 and 1 is needed to make 2 at 5/7

5/5=2 because non carry zone above a 1 is always 2

This is the Non carry above Carry above 2 scenario and is always 2 over 1 as shown.

Colum	0	1	2	3	4	5	6	7	8	9	10	11

Column	0	1	2	3	4	5	6	7	8	9	10	11
Row 0												
Row 1		1	1									
Row 2			2									
Row 3			1	0								
Row 4				1	2							
Row 5					2	2	IF X					
Row 6					1	1	IF 2					
Row 7						2	If 1					
Row 8						1	0					
Row 9							1					
10								2				
11								1	0			
12									1			
13										2		
14										1	0	
15											1	
16												
17												
18												
19												
20												

4/4=2 because if 1 it would end carry zone and 5/5=2 would be below non carry zone

IF 6/7=1 Then 6/6=2 This is because in a non carry zone the square above a 1 is always 2. But his is a violation because 6/6=2 means there is a 2 under the non carry zone at 6/5. This falsifies the IF case QED 6/7!=1 and must instead be 6/7=0

Colum	0	1	2	3	4	5	6	7	8	9	10	11

Column	0	1	2	3	4	5	6	7	8	9	10	11
Row 0												
Row 1		1	1	IF 2 X								
Row 2			2	IF 1								
Row 3			1	0	If 2							
Row 4				1	2							
Row 5					2	2						
Row 6					1	1						
Row 7						2	0					
Row 8						1	0					
Row 9							1					
10								2				
11								1	0			
12									1			
13										2		
14										1	0	
15											1	
16												
17												
18												
19												
20												

Location Column / Row

6/7=0 from last page.

If 4/3=2 Then 3/2=1 to block having non carry zone above 4/3=2

Then 3/1=2 because non carry above 1 always 2

But this makes 0/0=1 when we move BACKWARDS from row 1 to row 0 and this violates ABABBB being possible. QED 4/3=1

Column	0	1	2	3	4	5	6	7	8	9	10	11
Row 0												
Row 1		1	1	IF 0	IF X							
Row 2			2	IF 0	IF 2							
Row 3			1	0	1							
Row 4				1	2	1						
Row 5					2	2						
Row 6					1	1						
Row 7						2	0					
Row 8						1	0					
Row 9							1					
10								2				
11								1	0			
12									1			
13										2		
14										1	0	
15											1	
16												
17												
18												
19												
20												

Location Column / Row

4/3=1 from last page

IF 3/1=0 Then 3/2=0 via forward math 0 divided by 2 is 0

Then 4/2=2 because non carry above 1 is always 2 but this makes violation at 4/1

QED 3/1 !=0 and can't be 2 or 0/0=1 disproving ABABBB so 3/1=1

5/4=1 to avoid having 2 under non carry zone 5/3

Colum	0	1	2	3	4	5	6	7	8	9	10	11

Column	0	1	2	3	4	5	6	7	8	9	10	11
Row 0												
Row 1		1	1	1	If 0							
Row 2			2	0	X IF 1							
Row 3			1	0	1							
Row 4				1	2	1						
Row 5					2	2						
Row 6					1	1						
Row 7						2	0					
Row 8						1	0					
Row 9							1					
10								2				
11								1	0			
12									1			
13										2		
14										1	0	
15											1	
16												
17												
18												
19												
20												

Location Column / Row 3/1=1 from last page

IF 4/1=0 Then 4/2=1 because 0 in carry zone so 0+3 divided by 2 is 1

This makes a violation because 4/2 is non carry zone and 1 divided by 2 is 0 not the 1 needed at 4/3=1

QED 4/1 != 0

4/1!=2 because that would make 0/0=1 and disprove ABABBB so 4/1=1

Column	0	1	2	3	4	5	6	7	8	9	10	11
Row 0												
Row 1		1	1	1	1							
Row 2			2	0	2							
Row 3			1	0	1	2						
Row 4				1	2	1						
Row 5					2	2						
Row 6					1	1						
Row 7						2	0					
Row 8						1	0					
Row 9							1					
10								2				
11								1	0			
12									1			
13										2		
14										1	0	
15											1	
16												
17												
18												
19												
20												

Location Column / Row

4/1=1 from last page

4/2=2 because 1+3 carry from carry zone divided by 2 is 2

5/3=2 non carry above 1 always 2.

Violation 5/2 is a non carry zone and has 5/3=2 below it

This proves that ABABBB is impossible (because of the adjustments to keep it ABABBB) and will always be BBABBB!

Now Prove ABBABBB is impossible in forward direction

This segment is ONLY considering the possibility of ABBABBB segments existing. We will disprove this by showing that given BBABBB we must have BBBABBB despite trying to force ABBABBB.

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Row 0														
Row 1		1	1											
Row 2			2											
Row 3			1	0										
Row 4				1										
Row 5					2									
Row 6					1	0								
Row 7						1								
Row 8							2							
Row 9							1							
10								2						
11								1	0					
12									1					
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

3/3=0, 5/6=0, 8/11=0, 10/14=0, 12/17=0 because carry zone above 1 is 0

2/1=1 other wise 0/0 is 1 and BBBABBB instead of ABBABBB

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1											
2			2	0										
3			1	0										
4				1										
5					2									
6					1	0								
7						1	2							
8							2	2						
9							1	1						
10								2	0					
11								1	0					
12									1					
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

7/9=1 because non carry only permits 0 or 1 and carry zone needs 1 to make 2

7/8=2 because non carry above 1 always 2

6/7=2 because if 1 it would end carry zone & violation of 7/8=2 below non carry zone

3/2=0 because if it was 1 it would make 3/1=2 and that would make 0/0=1 -> BBBABBB

8/10=0 because 8/8 makes 8/9=0 or 1 and 8/9 makes 8/10=0 aka below 2 non carry

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1											
2			2	0	If X									
3			1	0	If 2									
4				1	If 1									
5					2									
6					1	0								
7						1	2							
8							2	2						
9							1	1						
10								2	0					
11								1	0					
12									1					
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

If 4/4=1 Then 4/3=2 and Violation at 4/2 because non carry zone 4/2 above 4/3=2

QED 4/4=2

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1											
2			2	0										
3			1	0										
4				1	2	0								
5					2	1								
6					1	0								
7						1	2							
8							2	2						
9							1	1						
10								2	0					
11								1	0					
12									1					
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

4/4=2 from previous page

5/5=1 to Avoid 0 under carry zone 5/4 which would be 0 under carry zone

5/4=0 because carry zone above 1 is always 0

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1										
2			2	0	2									
3			1	0	1									
4				1	2	0								
5					2	1								
6					1	0								
7						1	2							
8							2	2						
9							1	1						
10								2	0					
11								1	0					
12									1					
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

4/3=1 to end carry before 5/3 with 5/4=0 under it

4/2=2 because non carry above 1 always 2

3/1=1 to create carry zone so 4/2=2 is not under non carry zone

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0								
5					2	1								
6					1	0	2							
7						1	2							
8							2	2						
9							1	1						
10								2	0					
11								1	0					
12									1					
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

4/1=1 because can't be 2 or 0/0=1 and BBBABBB breaks scenario.

6/6=2 because it is below two carry zones at 6/4 and 6/5

5/3=0 because 2 non carry zones above. 5/2 can be 0 or 1 and makes 5/3=0

5/2=0 in non carry both 5/1=0 and 5/1=1 produce 0 below and 5/1=2 would make 0/0=1 and BBBABBB breaks scenario

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2						
9							1	1						
10								2	0					
11								1	0					
12									1					
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

6/4=0 because 2 non carry zones above it 6/2 limits 6/3 to 0 or 1 and 6/4 to 0

6/5=1 carry zone 0 always produces a 1 below it

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2						
9							1	1						
10								2	0					
11								1	0					
12									1	IF 2				
13										2	If 1			
14										1	0	IF 1		
15											1	IF 2	IF 0	
16												2	IF 1	
17												1	0	
18													1	
19														2
20														1

Location Column / Row

IF 9/12=2 Then 10/13=1 because can’t have 10/13=0 under carry zone 10/12

Then 11/15=2 because under two carry zones

Then 12/16=1 to avoid 0 under carry zone at 12/15,

Then 12/15=0 carry zone above 1 is always 0

Then 11/14=1 to close carry zone to avoid violation at 12/15=0 under carry zone 12/14

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2						
9							1	1						
10								2	0					
11								1	0					
12									1	IF 2	If 0	X		
13										2	If 1	IF 0		
14										1	0	IF 1		
15											1	IF 2	IF 0	
16												2	IF 1	
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 10/12=0 because carry zone above 10/13=1 is always 0

Then 11/13=0 because carry zone above 11/14=1 is always 0

There is a violation because Carry zone 11/12 is above 11/13=0!!

QED 9/12!=2 so 9/12=1

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2						
9							1	1						
10								2	0					
11								1	0	0				
12									1	1				
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

9/12=1 from previous page

9/11=0 inside carry above 9/12=1 always 0

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1	IF 2	If X					
6					1	0	2	IF 1	IF 2					
7						1	2	IF 2	IF 1					
8							2	2	IF 2					
9							1	1	IF 1					
10								2	0					
11								1	0	0				
12									1	1				
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

If 8/9=1 Then 8/8=2 because non carry above 1 is always 2

Then 7/7=2 because if it was a 1 it would end carry zone making 8/8=2 below non carry

Then 7/6=1 to avoid having 2 under non carry zone 7/5

Then 7/5=2, 8/6=2 because non carry above 7/6=1, 8/7=1 always 2

But now violation at 8/5 which is a non carry zone above a 2 at 8/6=2 QED8/9!=1

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2						
9							1	1	0					
10								2	0					
11								1	0	0				
12									1	1				
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

8/9=0 from previous page

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2	IF 0	If 0					
7						1	2	IF 1	IF 1					
8							2	2	If 0					
9							1	1	0					
10								2	0					
11								1	0	0				
12									1	1				
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

If 8/8=0 Then 7/7=1 to close carry zone before above 8/8=0

Then 7/6=0 because carry above 7/7=1 is always 0

Then 8/7=1 to avoid 0 under carry zone at 8/6

Then 8/6=0 because carry above 1 always 0;

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0	If X						
5					2	1	1	If 0						
6					1	0	2	IF 0	If 0					
7						1	2	IF 1	IF 1					
8							2	2	If 0					
9							1	1	0					
10								2	0					
11								1	0	0				
12									1	1				
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 7/5=0 above 7/6=0 and 8/6=0 if 7/5 was 1 it would create a carry zone over 8/6=0 which is a violation.

There is now a violation because 7/5=0 under a carry zone at 7/4

QED 8/8!=0 so 8/8=1

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1					
9							1	1	0					
10								2	0					
11								1	0	0				
12									1	1				
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

8/8=1 from last page

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1	If 0				
9							1	1	0	IF 1				
10								2	0	If 0	If 2			
11								1	0	0	IF 1			
12									1	1	IF 2			
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

IF 9/10=0 then 9/9=1 to block 9/9=0 under carry zone 9/8

Then 9/8=0 because carry zone above 9/9=1 is always 0

Then 10/10=2 because under 2 carry zones 10/8 and 10/9

Then 10/11=1 because non carry 10/10=2 divided by 2 is 1

Then 10/12=2 because carry zone 1 +3 divided by 2 is 2.

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1	If 0				
9							1	1	0	IF 1				
10								2	0	If 0	If 2			
11								1	0	0	IF 1			
12									1	1	IF 2	If 0		
13										2	If 1	If 0		
14										1	0	IF 1		
15											1	IF 2		
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 10/13=1 because 10/12 is non carry 2 divided by 2 is 1

Then 11/12=0 under 2 non carry zones 11/11 and 11/10 is always 0,

Then 11/13=0 because 0 divided by 2 is 0, 11/14=1 because 0 +3 divided by 2 is 1

Then 11/15=2 because carry above 1+3 divided by 2 is 2

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0	If 0						
5					2	1	1	IF 1						
6					1	0	2	IF 0						
7						1	2	IF 1						
8							2	2	1	If 0				
9							1	1	0	IF 1				
10								2	0	If 0	If 2			
11								1	0	0	IF 1			
12									1	1	IF 2	If 0		
13										2	If 1	If 0		
14										1	0	IF 1		
15											1	IF 2		
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 7/7=1 non local block of carry above 9/8=0 , 7/7 is only place with 2 below it that can have a carry zone closing 1 above.

Then 7/6=0 carry zone above 7/7=1 is always 0

Then 7/5=1 to block 7/5=0 under carry zone at 7/4,

Then 7/4=0 because carry zone above 7/5=1 is always 0

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0	If 0						
5					2	1	1	IF 1						
6					1	0	2	IF 0	If 2	If 0				
7						1	2	IF 1	IF 2	IF 1				
8							2	2	1	If 0				
9							1	1	0	IF 1				
10								2	0	If 0	If 2			
11								1	0	0	IF 1			
12									1	1	IF 2	If 0		
13										2	If 1	If 0		
14										1	0	IF 1		
15											1	IF 2		
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 8/6=2 because below 2 carry zones at 8/4 and 8/5

Then 8/7=2 because 8/6=2 in carry zone always makes 8/7=2,

Then 9/7=1, 9/7 cannot be 0 because below carry zone at 9/6

Then 9/6=0 because carry zone above 9/7=1 is always 0

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0	If 0						
5					2	1	1	IF 1						
6					1	0	2	IF 0	If 2	If 0				
7						1	2	IF 1	IF 2	IF 1				
8							2	2	1	If 0	If 2			
9							1	1	0	IF 1	IF 2			
10								2	0	If 0	If 2	IF 2		
11								1	0	0	IF 1	If 1		
12									1	1	IF 2	If 0		
13										2	If 1	If 0		
14										1	0	IF 1		
15											1	IF 2		
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 10/8=2 because below 2 carry zones 10/6 and 10/7

Then 10/9=2 because 10/8=2 in carry zone always produces 2 below it

Then 11/10=2 because below 2 carry zones 11/8 and 11/9

Then 11/11=1 because 11/10 is in non carry zone and 2 divided by 2 is 1

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0	IF 0							
3			1	0	1	0	IF 0	IF 0						
4				1	2	0	0	If 0	IF 0					
5					2	1	1	IF 1	IF 1					
6					1	0	2	IF 0	If 2	If 0				
7						1	2	IF 1	IF 2	IF 1				
8							2	2	1	If 0	If 2			
9							1	1	0	IF 1	IF 2			
10								2	0	If 0	If 2	IF 2		
11								1	0	0	IF 1	If 1		
12									1	1	IF 2	If 0		
13										2	If 1	If 0		
14										1	0	IF 1		
15											1	IF 2		
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 8/5=1 to close carry zone before above 9/6=0

Then 8/4=0 because carry above 8/5=1 always 0

Then 6/3=0 and 7/3=0, because 6/4=0, 7/4=0, 8/4=0 and any 1 before 8/3 would create a carry zone that had either 7/3=0 or 8/3=0 under it.

Then 6/2=0 because 6/3=0 and 7/3=0 and if 6/2=1 then carry zone above 7/3=0 violation

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1	IF 0								
2			2	0	2	0	IF 0							
3			1	0	1	0	IF 0	IF 0						
4				1	2	0	0	If 0	IF 0	If 0				
5					2	1	1	IF 1	IF 1	IF 1				
6					1	0	2	IF 0	If 2	If 0	IF 2			
7						1	2	IF 1	IF 2	IF 1	IF 2			
8							2	2	1	If 0	If 2	IF 2		
9							1	1	0	IF 1	IF 2	If 2		
10								2	0	If 0	If 2	IF 2		
11								1	0	0	IF 1	If 1		
12									1	1	IF 2	If 0		
13										2	If 1	If 0		
14										1	0	IF 1		
15											1	IF 2		
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 5/1=0, because 1 would start carry and violate because 6/2=0 below carry zone

Then 9/5=1 to avoid 9/5=0 under carry zone 9/4 above

Then 9/4=0 because carry zone above 9/5=1 is always 0

Then 10/6=2 below two carry zones above , 10/7=2 carry zone 2 always makes 2

Then 11/8=2 because two carry zones above , 11/9=2 carry zone 2 always makes 2

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1	IF 0								
2			2	0	2	0	IF 0							
3			1	0	1	0	IF 0	IF 0						
4				1	2	0	0	If 0	IF 0	If 0				
5					2	1	1	IF 1	IF 1	IF 1				
6					1	0	2	IF 0	If 2	If 0	IF 2			
7						1	2	IF 1	IF 2	IF 1	IF 2			
8							2	2	1	If 0	If 2	IF 2		
9							1	1	0	IF 1	IF 2	If 2		
10								2	0	If 0	If 2	IF 2	IF 2	
11								1	0	0	IF 1	If 1	IF 1	
12									1	1	IF 2	If 0	If 2	
13										2	If 1	If 0	If 1	
14										1	0	IF 1	If 2	
15											1	IF 2		
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 12/10=2 because two carry zones above, 12/11=1 because non carry 2 divided by 2=1

then 12/12=2, 12/11=1 in carry zone so 1+3 divided by 2 is 2

Then 12/13=1 because 2 divided by 2 is 1

Then 12/14=2 1 in carry is 1+3 divided by 2 equals 2

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1	IF 0								
2			2	0	2	0	IF 0							
3			1	0	1	0	IF 0	IF 0						
4				1	2	0	0	If 0	IF 0	If 0				
5					2	1	1	IF 1	IF 1	IF 1				
6					1	0	2	IF 0	If 2	If 0	IF 2			
7						1	2	IF 1	IF 2	IF 1	IF 2			
8							2	2	1	If 0	If 2	IF 2		
9							1	1	0	IF 1	IF 2	If 2		
10								2	0	If 0	If 2	IF 2	IF 2	
11								1	0	0	IF 1	If 1	IF 1	
12									1	1	IF 2	If 0	If 2	
13										2	If 1	If 0	If 1	
14										1	0	IF 1	If 2	
15											1	IF 2	IF 1	
16												2	IF 2	
17												1	0	
18													1	
19														2
20														1

Location Column / Row

Then 12/15=1 non carry 2 divided by 2 is 1

Then 12/16=2 because carry zone 1+3 divided by 2 is 2

But 12/16=2 cannot produce the 12/17=0 and we have a violation!

QED 9/10!=0

9/10=1 see next page

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1					
9							1	1	0					
10								2	0	1				
11								1	0	0				
12									1	1				
13										2				
14										1	0			
15											1			
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

9/10=1 from last page

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1					
9							1	1	0	2				
10								2	0	1				
11								1	0	0				
12									1	1	2			
13										2	1			
14										1	0			
15											1	2		
16												2		
17												1	0	
18													1	
19														2
20														1

Location Column / Row

9/9=2 non carry above 1 always 2

10/12=2 because below 2 carry zones 10/10 and 10/11

10/13=1 non carry zone 2 divided by 2 is 1

11/15=2 because below 2 carry zones

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1					
9							1	1	0	2				
10								2	0	1				
11								1	0	0				
12									1	1	2			
13										2	1			
14										1	0			
15											1	2		
16												2	IF 0	
17												1	0	If 0
18													1	If 1
19														2
20														1

Location Column / Row

IF 13/18=1 then 13/17=0 because carry zone above 13/18=1 always 0

Then 12/16=0 because 1 would open carry zone above 13/17=0

But 12/16=0 is a violation 0 under carry zone at 12/15

QED 13/18!=1

13/18=2

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1					
9							1	1	0	2				
10								2	0	1				
11								1	0	0				
12									1	1	2			
13										2	1			
14										1	0	1		
15											1	2	0	
16												2	1	
17												1	0	
18													1	2
19														2
20														1

Location Column / Row

13/18=2 from last page

12/16=1 because below carry so can only be 1 or 2 and only 1 works for 0 below

12/15=0 carry above a 1 is always 0

11/14=1 to close carry zone before above the 0 at 12/15

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1					
9							1	1	0	2				
10								2	0	1				
11								1	0	0				
12									1	1	2	0		
13										2	1	0	0	
14										1	0	1	1	
15											1	2	0	
16												2	1	
17												1	0	
18													1	2
19														2
20														1

Location Column / Row

11/13=0 because in carry zone and above 11/14=1

12/14=1 to avoid 12/14=0 under carry zone at 12/13

12/13=0 because in carry zone and above 12/14=1

11/12=0 because if 1 there would be carry zone at 11/13 above 12/13=0

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1					
9							1	1	0	2				
10								2	0	1	0	0		
11								1	0	0	1	1		
12									1	1	2	0		
13										2	1	0	0	
14										1	0	1	1	
15											1	2	0	
16												2	1	
17												1	0	
18													1	2
19														2
20														1

Location Column / Row

10/11=1 to end carry zone before above 11/12=0

10/10=0 carry zone above 10/11=1 is always 0

11/11=1 to avoid 11/11=0 under carry zone at 11/10

11/10=0 because carry zone above 1 is always 0

Colu	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1		1	1	1	1									
2			2	0	2	0								
3			1	0	1	0								
4				1	2	0	0							
5					2	1	1							
6					1	0	2							
7						1	2							
8							2	2	1					
9							1	1	0	2	0			
10								2	0	1	0	0	X	
11								1	0	0	1	1	0	
12									1	1	2	0	1	
13										2	1	0	0	
14										1	0	1	1	
15											1	2	0	
16												2	1	
17												1	0	
18													1	2
19														2
20														1

Location Column / Row

10/9=0 because if it was 1 it would open up carry zone above 11/10=0

12/12=1 to avoid having 0 under carry zone at 12/11

12/11=0 because carry zone above 12/12=1 always 0,

12/10 is in violation because it is a carry zone above a 0 at 12/11!!!!!!!

QED ABBABBB is impossible!

Significance of what is proven so far

We have shown that the sequence ABB descends and therefore another B must be added and there are no ways to add a B to ABBABBABB……ABB without having a segment of BBB.

The segment BBB once it exists proves that a segment ABBB must exist because if BBBB is encountered we can just shift over and look at the letter before and consider that our ABBB.

We have proven that once we find ABBB that it cannot be AABBB so we must have BABBB.

We have then shown that ABABBB in fact cannot exist because we always get BBABBB.

Finally we have proven that ABBABBB does not exist and always indicates BBBABBB.

The problem here is that ABBB ascends and must touch a segment that descends but every time we find ABBB we find BBBABBB and we can then start the process over with BBB(ABBB) looking leftward for an A. As we move left ward through the loop all the segments that are expelled on the right side ascend. (ABBB)(B)(ABBB)……ETC.

This forms a paradox because ABBABB…..ABB always descends but as soon as we consider a case with more Bs than ABB would give us we get a case that is infinitely ascending. We cannot add a fractional part of B=121 so we are left with a proof of infinite descent ABB or infinite ascent when BBB exists.

Let me restate that this logic only works for loops or segments that have a starting and ending digit length that is identical. This logic does not work in the general case and I use another method of the general non loop case to argue that infinite ascent is unlikely.

END of Loop Proof

Start of sequence “almost” proof

Consider #01 and #2

In a loop there must always be a #2 number or a #01 number somewhere in the loop if it has 3 steps or more. Here # represents the larger digits and 2 is the least significant digit not to be confused with 2# where # was the smaller digits.

Sub-Proof

Because of the 3n+1 and divide by 2 the right most digit can never be 0. This is because after a 3n+1 moving forwards the digit is always 1. Dividing the 1 produces a 2 and dividing the 2 again provides a 1. There is no pathway back to a 0. Because the sequence is infinite just advance past the first shift and name that new number the "start" of the sequence.

QED #0 base 3 cannot be the right most digit.

Now lets eliminate all #2 numbers as they satisfy our claim already. Consider all possible 2 digit rightmost digits remaining after removing #01 which satisfies our claim:

#11 and #21 remain.

Consider #11 base 3:

If the next operation moving backwards is multiply then #11->#22 satisfies the #2 case.

If the next operation is unshift then the move afterwards must be a multiply.

#11->#1->#2 Which satisfies the case #2.

Consider #21 base 3:

If the next operation moving backwards is multiply then #21->#12 satisfies the #2 case.

If the next operation is unshift then #21->#2 satisfies the case #2.

QED a loop of more than 3 steps always contains either #2 or #01.

Branching Backwards

In considering an unknown number we don't know if the number is even or not and the expectation is that every step forward will branch into 2 possible numbers due to uncertainty on wether the number is even or odd. Surprisingly taking steps backwards in time branches far less frequently. This becomes clear when we consider things that are impossible moving backwards.

First

#0

Is an impossible value except for the initial seed number that starts a sequence. We have agreed to advance past the first 3n+1 and call this the seed number to eliminate #0 as a possibility.

Second

#N1

Unshifts to

#N

But

#N2

Cannot unshift because shifts always put a 1 in the right most digit and a 2 means the previous operation must have been a multiply/divide.

Third

#N01

Cannot unshift because to do so would put a 0 in the right most position.

Again this is true for loops and all but the first number of a infinite sequence(which can be converted via stepping past the first 3n+1 into sequence that always obeys this rule.

Fourth

It is impossible to have two shifts back to back. This is because 3n+1 is always even so moving forward the next operation must be a divide and because 3n+1 is not a divide the operation before it cannot be 3n+1.

Examples:

#02

Must be a reverse divide because it cannot be a shift as right most value is not 1.

Because we are doing a reverse division we can multiply from the right by 2

2*2=4 which is 11 in base 3.

So we can see the previous value was

#11

Consider

#101

Here we have a 1 on the right most digit and it may look like we can reverse shift but the 0 makes that impossible because we would have a 0 in the right most digit. So we must multiply by 2.

#202

Because of the right most 2 we cannot unshift and must multiply by 2

#111

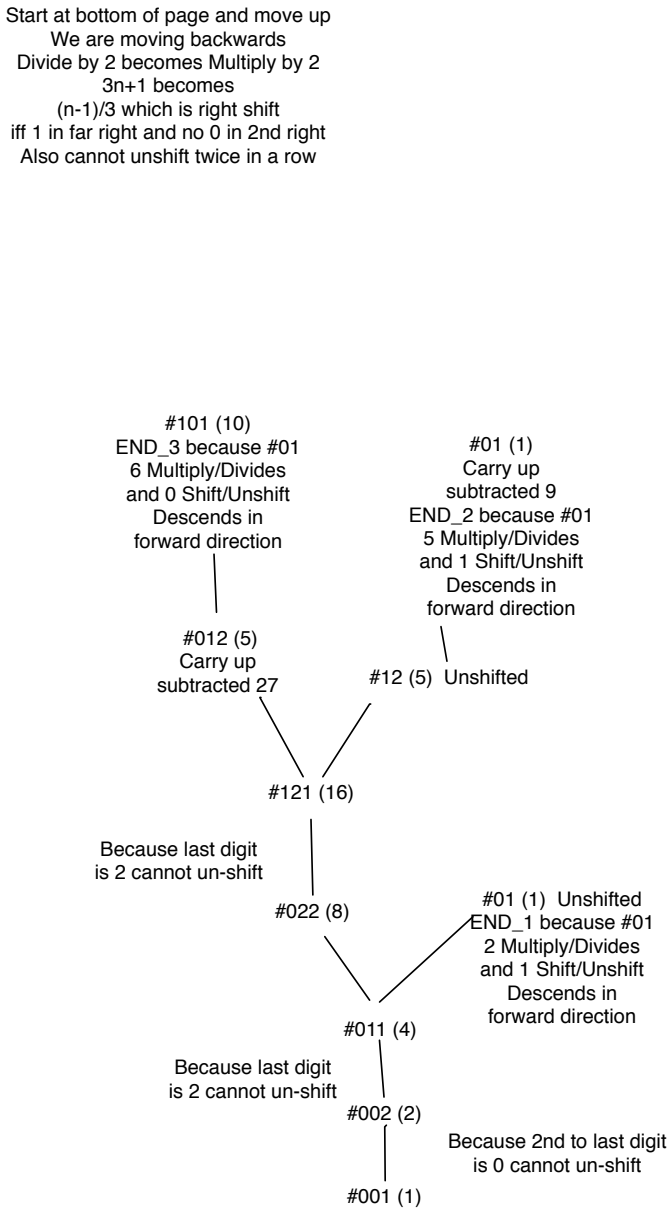
This is the end of examples

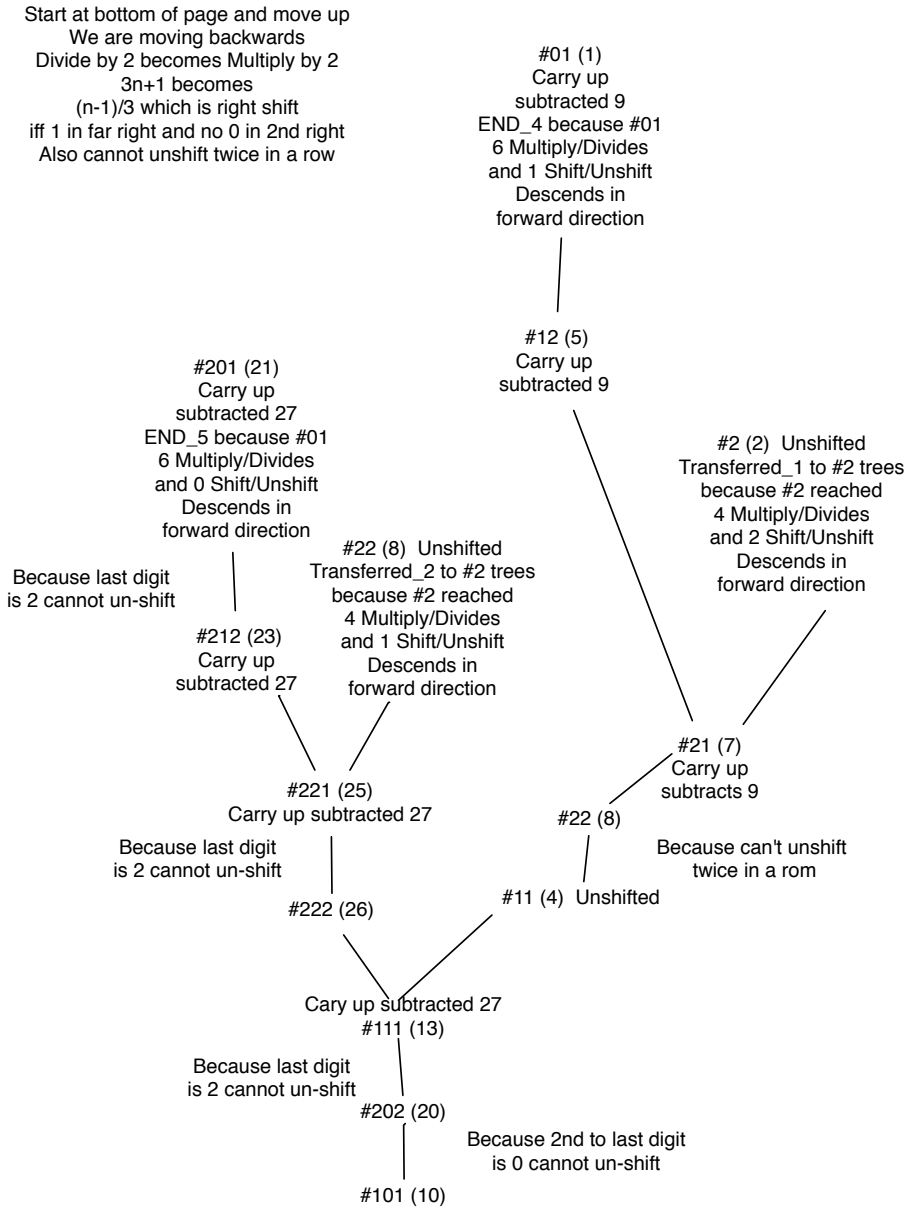
All but One Path Descend

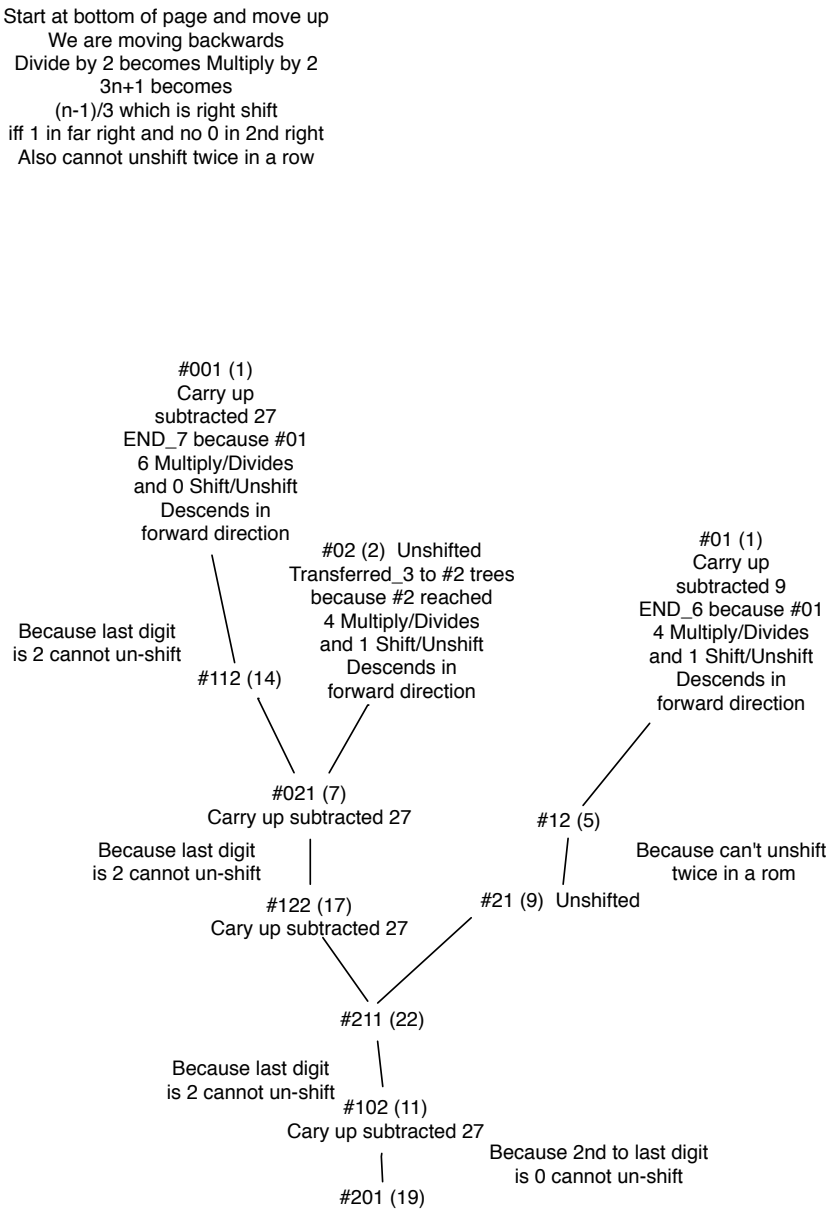
Consider the examples below were all the pathways except One descend when stepping in the positive direction. #01 is replaced by #001, #101 or #201 and #2 is replaced by #02, #12, or #22.

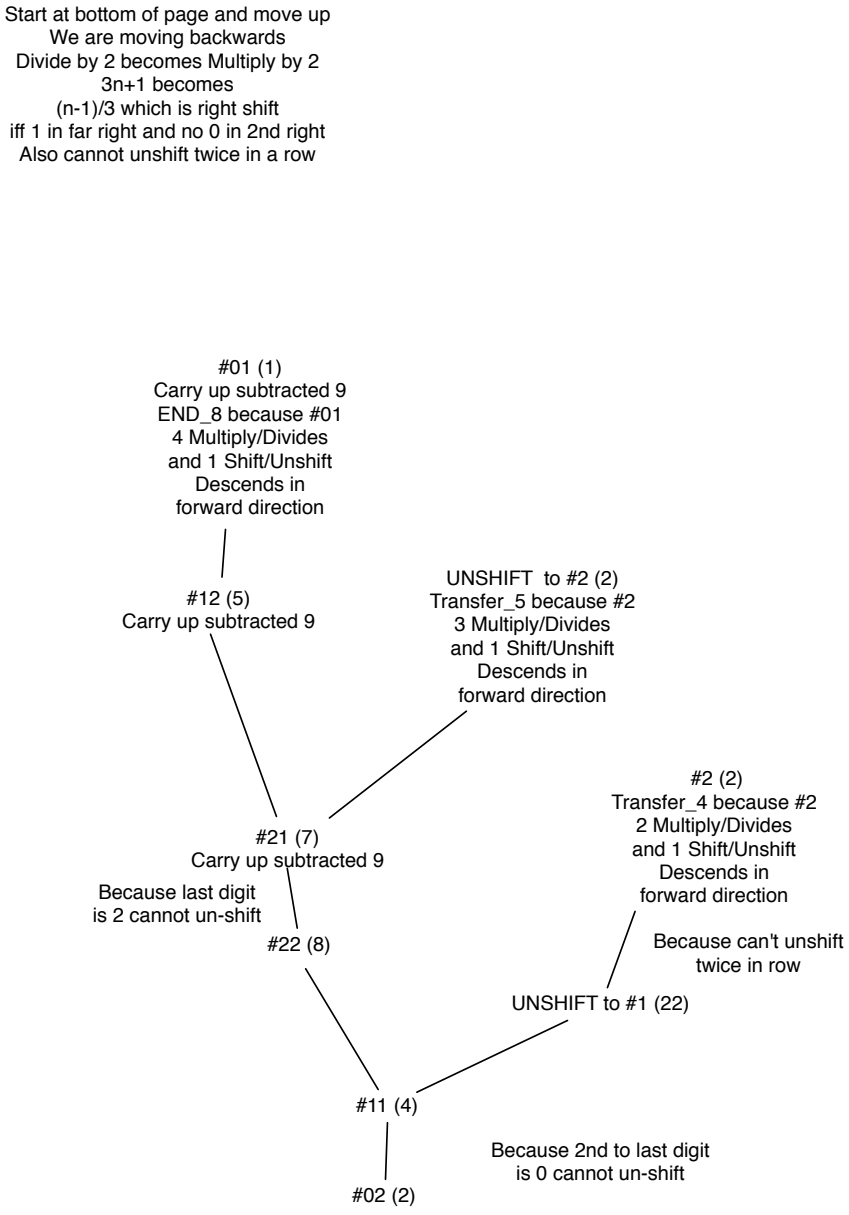
The trees however are created by stepping in the negative time direction so that multiply by 2 and (n-1)/3 are used along with the logic described in branching backwards.

To help show this the move trees will start at the bottom and extend backwards so that they can easily be checked by stepping forwards from top to bottom.




Start at bottom of page and move up
We are moving backwards
Divide by 2 becomes Multiply by 2
3n+1 becomes
(n-1)/3 which is right shift
iff 1 in far right and no 0 in 2nd right
Also cannot unshift twice in a row
#01 (1)
Carry up
subtracted 9
END_4 because #01
6 Multiply/Divides
and 1 Shift/Unshift
Descends in
forward direction
#12 (5)
Carry up
subtracted 9
#201 (21)
Carry up
subtracted 27
END_5 because #01
6 Multiply/Divides
and 0 Shift/Unshift
Descends in
forward direction
#2 (2) Unshifted
Transferred_1 to #2 trees
because #2 reached
4 Multiply/Divides
and 2 Shift/Unshift
Descends in
forward direction
Because last digit
is 2 cannot un-shift
#22 (8) Unshifted
Transferred_2 to #2 trees
because #2 reached
4 Multiply/Divides
and 1 Shift/Unshift
Descends in
forward direction
#212 (23)
Carry up
subtracted 27
#21 (7)
Carry up
subtracts 9
#221 (25)
Carry up subtracted 27
#22 (8)
Because last digit
is 2 cannot un-shift
Because can't unshift
twice in a rom
#222 (26)
#11 (4) Unshifted
Cary up subtracted 27
#111 (13)
Because last digit
is 2 cannot un-shift
#202 (20)
Because 2nd to last digit
is 0 cannot un-shift
#101 (10)


Start at bottom of page and move up
We are moving backwards
Divide by 2 becomes Multiply by 2
3n+1 becomes
(n-1)/3 which is right shift
iff 1 in far right and no 0 in 2nd right
Also cannot unshift twice in a row
#001 (1)
Carry up
subtracted 27
END_7 because #01
6 Multiply/Divides
and 0 Shift/Unshift
Descends in
forward direction
#02 (2) Unshifted
Transferred_3 to #2 trees
because #2 reached
4 Multiply/Divides
and 1 Shift/Unshift
Descends in
forward direction
#01 (1)
Carry up
subtracted 9
END_6 because #01
4 Multiply/Divides
and 1 Shift/Unshift
Descends in
forward direction
Because last digit
is 2 cannot un-shift
#112 (14)
#021 (7)
Carry up subtracted 27
#12 (5)
Because last digit
is 2 cannot un-shift
Because can't unshift
twice in a rom
#122 (17)
Cary up subtracted 27
#21 (9) Unshifted
#211 (22)
Because last digit
is 2 cannot un-shift
#102 (11)
Cary up subtracted 27
Because 2nd to last digit
is 0 cannot un-shift
#201 (19)


Start at bottom of page and move up
We are moving backwards
Divide by 2 becomes Multiply by 2
3n+1 becomes
(n-1)/3 which is right shift
iff 1 in far right and no 0 in 2nd right
Also cannot unshift twice in a row
#01 (1)
Carry up subtracted 9
END_8 because #01
4 Multiply/Divides
and 1 Shift/Unshift
Descends in
forward direction
#12 (5)
Carry up subtracted 9
UNSHIFT to #2 (2)
Transfer_5 because #2
3 Multiply/Divides
and 1 Shift/Unshift
Descends in
forward direction
#2 (2)
Transfer_4 because #2
2 Multiply/Divides
and 1 Shift/Unshift
Descends in
forward direction
#21 (7)
Carry up subtracted 9
Because last digit
is 2 cannot un-shift
#22 (8)
Because can't unshift
twice in row
UNSHIFT to #1 (22)
#11 (4)
Because 2nd to last digit
is 0 cannot un-shift
#02 (2)

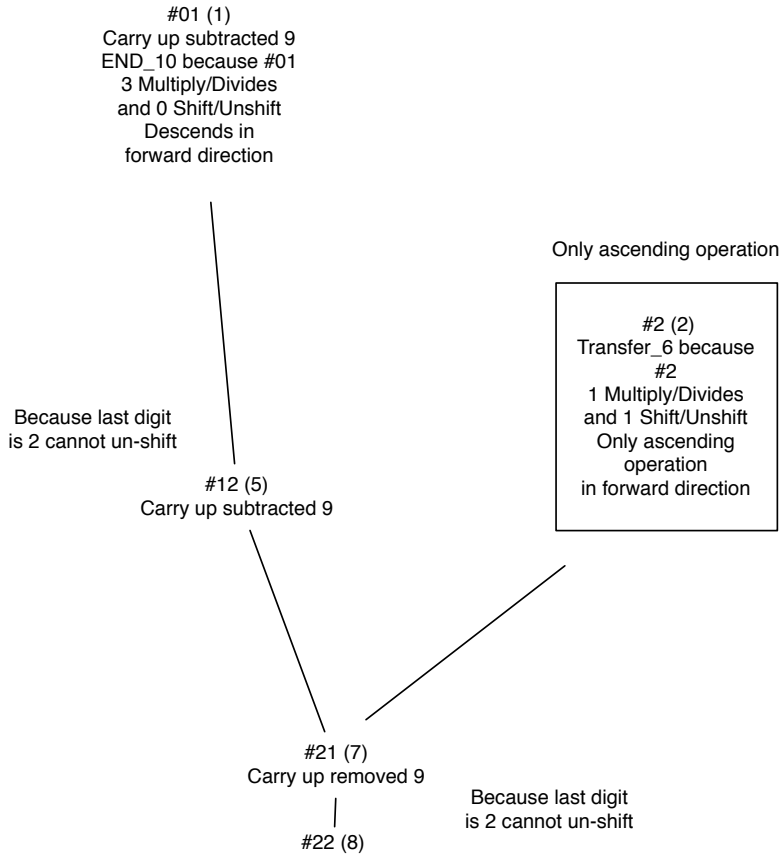
Start at bottom of page and move up
We are moving backwards
Divide by 2 becomes Multiply by 2
3n+1 becomes
(n-1)/3 which is right shift
iff 1 in far right and no 0 in 2nd right
Also cannot unshift twice in a row

#01 (1)
Carry up subtracted 9
END_9 because #01
1 Multiply/Divides
and 0 Shift/Unshift
Descends in
forward direction

Because last digit
is 2 cannot un-shift

#12 (5)

Start at bottom of page and move up
We are moving backwards
Divide by 2 becomes Multiply by 2
3n+1 becomes
(n-1)/3 which is right shift
iff 1 in far right and no 0 in 2nd right
Also cannot unshift twice in a row



Using this method we can see that

#2->#1->#11->#22 only works if the start is 3#+2 and is the only sequence that ascends.

Lets consider this from a different angle

ConsiderBase 4

We will consider all the odd numbers just before 3n+1 for 2 digits of base 4 numbers each digit consisting of 0,1,2,3. Because the number is odd 1 or 3 must be the least significant digit as 0 and 2 are even.

Seed number in base 4	Carry up	Result of first 3n+1	Next seed number	Descends	Fed by
#01	0	#10	#01 or #11 or #21 or #31	Yes unconditionally	#01 *1 of 4, #31*1 of 4, #23*1 of 2, #11 1 of 8
#11	1	#00	Next 2 digits if even more divides will occur until odd	Yes unconditionally	#01 *1 of 4, #31*1 of 4, #03*1 of 2, #11 1 of 8
#21	1	#30	#03, #13, #23 or #33	Yes unconditionally	#01 *1 of 4, #31*1 of 4, #23*1 of 2, #11 1 of 8
#31	2	#20	#01, #11, #21, or #31	Yes unconditionally	#01 *1 of 4, #31*1 of 4, #03*1 of 2, #11 1 of 8
#03	0	#22	#11 or #31 if above odd	Yes unconditionally when the next shift is considered	#21 *1 of 4, #13*1 of 2, #11 1 of 8
#13	1	#12	#03 or #23 if above even	Only if the number above is Odd prior to the shift	#21 *1 of 4, #03*1 of 2, #11 1 of 8
#23	2	#02	#01 or #21 if above odd	Only if the number above is even prior to the shift	#21 *1 of 4, #13*1 of 2, #11 1 of 8
#33	2	#32	#13 or #33 if above odd	No Unconditionally	#21 *1 of 4, #33*1 of 2, #11 1 of 8

Lets consider a few things:

1) #01, #11, #21, #31 all descend because 1 shift and 2, 4, 2, 3 divides respectively.

2) The average of all 8 possibilities descends as there are 8 shifts and at least 15 Divides. Using 3.001 for 3n+1 this is multiply by 6,578.52 vs divide by 32,768.

3) #03->#11->#00 or #03->#31->#20 both descend unconditionally with 4 or 5 divides and 2 shifts .

4) #13->#03->#11->#00 or #13->#03->#31->#20 descends conditionally if #03 path is taken with 5 or 6 divides and 3 shifts

5) #23->#01->#01, #11, #21 or #31->#X0 descends conditional if the #01 path is taken with 5 or more divides and 3 shifts

6) 5/8 pathways unconditionally descend, 2/8 pathways conditionally descend based on the number above being even or odd. The average descends. The numbers seem to rotate so that even though #33 does loop to itself in a stronger way that #01 loops to itself the net effect of #01 and #31 both looping to themselves half as often as #33 may even it out.

7) The only unconditionally ascending pathway is dependent on the polarity of the number above to maintain being #33 and when this is lost does not quickly return to this pattern but appears to visit many if not all of the other branches first.

This is not a proof that infinite ascent is impossible. Such a proof may not exist. But once the loop option has been removed it appears that infinite ascent is exceedingly unlikely.