

Collatz Conjecture, A Visual Exploration

Vida Vakilotojar, September 2021

Introduction

- The Collatz conjecture, also called the $3n+1$ conjecture, states that for any given positive integer n , if the following two rules are repeatedly applied, then we eventually reach the number 1, at which point we stop applying the rules:

If the current number n is divisible by 2,
then replace it with $n/2$

Otherwise, replace the number with $3*n + 1$,
and repeat.

Simplifications

- Given any positive integer n , consider the following while loop:

While n is not equal to 1:

 If there exists a largest integer $m \geq 1$ such that n is divisible by 2^m , then replace n with $n / (2^m)$

 Otherwise, replace n with $(3n + 1) / 2$

- The Collatz conjecture is equivalent to asking whether the above loop always halts.
- Note that in the above while loop, successive applications of the $n/2$ rule have been collapsed into a single rule. Also, since for any odd number n , $3n+1$ is always an even number, any application of the $(3n+1)$ rule and the inevitable application of the very next $n/2$ rule are combined into a single application of $(3n+1)/2$ rule.
- In this presentation, the focus is mainly on this modified version of the Collatz algorithm.

Considering Base 3

- Any positive integer n belongs to one and only one of the three following sets:
 - S_0 : contains odd numbers that are a multiple of 3, and any multiples of 2 of such numbers. Thus a number in S_0 has the general form: $2^p(3k)$, where k is odd.
 - S_1 : contains odd numbers that are of the form $3k + 1$ (where k is even), and any multiples of 2 of such numbers. Thus a number in S_1 has the general form: $2^p(3k+1)$, where k is an even number.
 - S_2 : contains odd numbers that are of the form $3k + 2$ (where k is odd), and any multiple of 2 of such numbers. Thus a number in S_2 has the general form: $2^p(3k+2)$, where k is an odd number.

Reachability of numbers in S_0

- A number in S_0 is either odd and of the form $3(2k+1)$, or is even and of the form $2^p(3(2k+1))$, where $p \geq 1$.
- Consider an even number in S_0 , of the simplified form $m = 6k$. Since this number is even, the $n/2$ rule applies to it, and thus after removal of all its factors of 2, it descends to a number of the form $m' = 3(2k'+1)$, where $m' < m$.
- Consider an odd number in S_0 , of the form $m = 3(2k+1)$. By the application of the $(3n+1)/2$ rule to this number, we reach $m' = (3(3(2k+1)) + 1)/2 = (18k + 10)/2 = 9k+5$. m' is a number in S_2 , and we have $m' > m$. That is, all odd S_0 numbers reach a number in S_2 .
- However if k itself is odd, then $m' = 9k+5$ will be even and further divisible by 2, and leading to a smaller number $m'' < m$. More accurately, if an odd number m in S_0 is of the form $m = 3(2(2k'+1)+1) = 12k'+9$, then the application of $(3n+1)/2$ will take us to $m' = (3(12k'+9)+1)/2 = (36k'+28)/2 = 18k'+14$, and then application of $n/2$ rule will take us to $m'' = m'/2 = 9k'+7$ which is in S_1 and $m'' < m$. But if an odd number m in S_0 is of the form $m = 12k'+3$, then application of $(3n+1)/2$ will take us to $m' = (3(12k'+3)+1)/2 = (36k'+10)/2 = 18k'+5$ that is in S_2 , and $m' > m$.
- If a number $m = 3k$ in S_0 were to be reachable by the application of $(3n+1)/2$ rule from another number m' , we would need to have $m = 3k = (3m'+1)/2$, or $6k - 1 = 3m'$. However, there is no solution for the latter equation, which means numbers in S_0 are not reachable by application of the Collatz rules. However, one can always start from a number in S_0 , and explore numbers that are reachable from it.

Reachability of numbers in S1

- A number m in $S1$ is either odd and of the general form $m = 3(2k)+1 = 6k+1$, or is even and of the form $m = 2^p(6k+1)$, (for any integer k).
- Consider an even number in $S1$ of the form $m = 2^p(6k+1)$. These numbers reach down to $m' = 6k+1$ after (repeated) applications of the $n/2$ rule.
- For an even number in $S1$ to be reachable by the $(3n+1)/2$ rule, we need to have $m = 2^p(6k+1) = (3m'+1)/2$, or that $2^{(p+1)}(6k+1) - 1 = 3m'$, which requires that $2^{(p+1)}(6k) + 2^{(p+1)} - 1$ be divisible by 3, which requires that $2^{(p+1)} - 1$ be divisible by 3. This equation always has a solution for m' if p is odd, and otherwise does not have a solution. Thus even numbers in $S1$ of the form $m = 2^p(6k+1)$, where p is odd, are reachable from smaller numbers m' via the application of Collatz rule $(3n+1)/2$, and once such m is reached, the $n/2$ rule becomes applicable, leading them to $m'' = m/2 < m'$.
- Consider an odd number in $S1$, of the form $m = 6k+1$. By the application of the $(3n+1)/2$ rule to this number, we reach $m' = (3(6k+1) + 1)/2 = (18k + 4)/2 = 9k+2$, which is a number in $S2$ and $m' > m$.
 - If k itself is even (i.e., $k = 2k'$, and thus $m = 12k'+1$), then by further application of the $n/2$ rule we reach $m'' = m'/2 = 9k'+1$, which is a number in $S1$, and $m'' < m$. Note that m'' may still be an even number and further divisible by 2.
 - If on the other hand k is odd (i.e., $k = 2k'+1$, and $m = 6(2k'+1)+1 = 12k'+7$), then $m' = (3n+1)/2 = 9k+2 = 9(2k'+1)+2 = 18k' + 11$, which is an odd number that is in $S2$, and $m' > m$.
- If an odd number in $S1$, of the form $m = 6k+1$ were to be reachable by the application of $(3n+1)/2$ rule from another number m' , we would need to have $m = 6k+1 = (3m'+1)/2$, or $12k + 1 = 3m'$. However, there is no solution for the latter equation, which means odd numbers in $S1$ are not reachable from smaller numbers by the application of the Collatz rules. However, one can always start from an odd number in $S1$, and explore numbers that are reachable from it.

Reachability of numbers in S2

- Any number in S2 is either odd and of the form $3(2k+1)+2 = 6k+5$, or is even and of the general form $2^p(6k+5)$.
- Consider an even number in S2 of the form $m = 2^p(6k+5)$. These numbers reach down to $m' = 6k+5$ after (repeated) applications of the $n/2$ rule.
- For an even number in S2 to be reachable by the $(3n+1)/2$ rule, we need to have $m = 2^p(6k+5) = (3m'+1)/2$, or that $2^{(p+1)}(6k+5) - 1 = 3m'$, which requires that $2^{(p+1)}(6k) + 2^{(p+1)}5 - 1$ be divisible by 3, which requires that $2^{(p+1)}5 - 1$ be divisible by 3. This equation always has a solution for m' if p is even, and otherwise does not have a solution. Thus even numbers in S2 of the form $m = 2^p(6k+5)$, where p is even, are reachable from smaller numbers m' via the application of Collatz rule $(3n+1)/2$, and once such m is reached, the $n/2$ rule becomes applicable, leading them to $m'' = m/2 < m'$.
- Consider an odd number in S2, of the form $m = 6k+5$. By the application of the $(3n+1)/2$ rule to this number, we reach $m' = (3(6k+5) + 1)/2 = (18k + 16)/2 = 9k+8$, which is a number in S2 and $m' > m$.
 - If k itself is even (i.e., $k = 2k'$, and thus $m = 12k'+5$), then by further application of the $n/2$ rule we reach $m'' = m'/2 = 9k'+4$, which is a number in S1, and $m'' < m$. Note that m'' may still be an even number and further divisible by 2.
 - If on the other hand k is odd (i.e., $k = 2k'+1$, and $m = 6(2k'+1)+5 = 12k'+11$), then $m' = (3n+1)/2 = 9k+8 = 9(2k'+1)+8 = 18k' + 17$, which is an odd number that is in S2, and $m' > m$.
- If an odd number in S2, of the form $m = 6k+5$ were to be reachable by the application of $(3n+1)/2$ rule from another number m' , we would need to have $m = 6k+5 = (3m'+1)/2$, or $12k + 9 = 3m'$, or that $4k+3 = m'$. This equation always has a solution for m' , where m is also odd, which means odd numbers in S2 are reachable from smaller numbers by the application of the Collatz rules. One can also always start from an odd number in S2, and explore numbers that are reachable from it.

Considering Base 12

- We can also consider all numbers in base 12. To simplify the analysis, ignore all even numbers, as after repeated application of the $n/2$ rule, they degenerate into an odd number.
- In base 12, we would be interested in 6 class of odd numbers.
 - If $n \% 12 == 1$, then n is in S_1 , and after the application of the $(3n+1)/2$ rule followed by the $n/2$ rule, we reach a smaller number in S_1 that maybe further divisible by 2. $n' = (3(12k+1)+1)/2 = (36k+4)/2 = 18k+2$, $n'' = n'/2 = 9k+1$
 - If $n \% 12 == 3$, then n is in S_0 , and after the application of the $(3n+1)/2$ rule, we reach a larger odd number in S_2 . $n' = (3(12k+3)+1)/2 = (36k+10)/2 = 18k+5$
 - If $n \% 12 == 5$, then n is in S_2 , and after the application of $(3n+1)/2$ rule followed by the $n/2$ rule, we reach a smaller number in S_1 that maybe further divisible by 2. $n' = (3(12k+5)+1)/2 = (36k+16)/2 = 18k+8$, $n'' = n'/2 = 9k+4$
 - If $n \% 12 == 7$, then n is in S_1 , and after the application of the $(3n+1)/2$ rule, we reach a larger odd number in S_2 . $n' = (3(12k+7)+1)/2 = (36k+22)/2 = 18k+11$
 - If $n \% 12 == 9$, then n is in S_0 , and after the application of $(3n+1)/2$ rule followed by the $n/2$ rule, we reach a smaller number in S_1 that maybe further divisible by 2. $n' = (3(12k+9)+1)/2 = (36k+28)/2 = 18k+14$, $n'' = n'/2 = 9k+7$
 - If $n \% 12 == 11$, then n is in S_2 , and after the application of the $(3n+1)/2$ rule, we reach a larger odd number in S_2 . $n' = (3(12k+11)+1)/2 = (36k+34)/2 = 18k+17$

Possible Proof Strategies

- One can try to prove the conjecture by proving that starting from any positive number $n > 1$, we can eventually reach a number that is smaller than the original number. Then by induction, we can prove that all numbers reach 1. We already know that all even numbers, and also odd numbers that modulo 12 are congruent with 1, 5, or 9 immediately reach a smaller number. The rest of numbers however reach a larger odd number is S_2 after one application of the $(3n+1)/2$ rule. S_2 numbers themselves are either 5 or 11 (modulo 12), and those in the former set (5) go down to a smaller number in S_1 , while the latter ones (11) go to still a larger odd number in S_2 . However, we also know that any odd number is S_2 is reachable from a smaller number. Thus, if there is any counterexample, the smallest of which cannot be in S_2 . Enumerating all the ways the numbers can move up and down, and formally proving that they all descend to 1 using number theoretic approaches, and for all numbers of infinitely large sizes has obviously been a big challenge.

Generating All The Numbers

- Alternatively, one can start from the number 1, and generate all the numbers that are reachable from it by applying the inverse of the Collatz rules, and then try to prove that all positive numbers are reachable. However, this problem does not seem to make a proof any easier.
- To generate all the numbers reachable, follow the steps bellow:

Have a frontier set of numbers that need to be explored, initialized with {1}. Also have a reachable set of numbers initialized as empty.

While the frontier set is not empty:

remove a number n from the frontier set, add $2n$ to the frontier set, and if $2n-1$ is divisible by 3 then also add $(2n-1)/3$ to the frontier set, and finally add n to the reachable set.

Finding a Counter Example

- One can try to search for a counterexample. If there exists any counterexample, there must be a smallest such counterexample, called Z . No number that is reachable from Z can be smaller than Z . Moreover, no number that reaches Z , or any number reachable from Z , can be smaller than Z . That is, if you consider any number Y reachable from Z , and from any such Y try to generate all numbers that reach Y (via the process described previously), none of the numbers can be smaller than Z . The reason is that any number that is smaller than Z is presumably one that can reach all the way down to 1, and thus if on their descend to 1 they reach Z or any number on the trajectory of Z , then Z must be reaching 1 itself.
- Note that the smallest counterexample cannot be in S_2 , because we have shown any odd number in S_2 can be reached from a smaller number.
- Proving that there does not exist any such Z , such that everything reachable from it is also larger than Z , and anything that reaches it and its trajectory is also larger than Z , is as hard, and while it may be hard to imagine that a single number and all the other numbers associated with it may have such a different behavior than all numbers smaller than Z , one can also imagine that such numbers may either infinitely grow larger, or have another attractor that is equal or larger than Z , just like 1 is an attractor for all numbers smaller than Z .

Our Approach

- While I have spent some time looking at other rather trivial approaches outlined above, I have found them quite unyielding and leading to nothing but exponentially growing proofs, as if to prove every number reaches 1, one has to effectively run the whole algorithm until 1 is reached. But of course the challenge is that one does not know if the algorithm would eventually halt, and if it keeps going on forever, you won't know if it means that 1 is not reachable for that instance, or if it would take even longer.
- The approach that I have taken does not conform to a formal and mathematical proof, but still provides strong evidence that there is a definite strong bias for all numbers to eventually descend to 1, simply because of the way the $(3n+1)/2$ manipulates the binary representation of the number, and thus fundamentally its structure, in a way that is independent of the starting number, and in a way that resembles an inevitable machinery that after a while of running and grinding gets into a homogeneous pattern of transitions that inevitably leads the state of the engine to reach 1.

Architecture of the Approach

- Basics of binary math:

$$\begin{aligned} 3 * n + 1 &= 2 * n + n + 1 = (n \ll 1) + n + 1 = ((n \ll 1) + 1) + n \\ n / 2 &= n \gg 1 \end{aligned}$$

- That is, to compute $3*n + 1$, you can shift the binary representation of the given number n to the left, flip the LSB bit of the result to 1, and then add the result to the original binary representation of the number.
- And to compute $n/2$, you can shift the binary representation of the given number n to the right.
- It turns out that if we use the binary representation of numbers, we would notice a lot of regular patterns and structure as the Collatz rules $(3n+1)/2$ are repeatedly applied to any given number and as the number descends to 1.
- Observe that a randomly selected binary numbers may look like a random sequence of 0s and 1s, but it can always be broken down into smaller segments (cells) that have more regular structure. The only regular patterns of interest for this analysis are ALL_ONES, ALL_ZEROS, and ALT_01S. And any given binary number is segmented from the right (LSB) to the left (MSB), one cell at a time. The following regular expressions are used to find a unique segmentation of any binary number. Note that when searching for a match at the LSB, the following expressions must be applied in the exact order specified; i.e., an ALT_10S rule would match only if neither the ALL_ONES, nor the ALL_ZEROS rule had succeeded. The matched cells must be of maximal length.

```
ALL_ONES   = r'1+$'  
ALL_ZEROS  = r'0+$'  
ALT_10S    = r'(10)+$'
```

Cell Transitions Reveal Problem Structure

- The $n/2$ rule simply shifts out the right most 0(s) of an even number while preserving the other bits. The $n/2$ rule (or its back to back applications) may remove an ALL_ZEROS cell at the right end of the number, or it may remove the rightmost bit of a ALT_10S cell at the right end of the number and thus break down that cell. However, the $n/2$ does not affect any other cells in the number.
- On the other hand, the $3n+1$ rule causes local changes in each individual cell in the number, that depend on the type of the cell, the bit on its left, and whether or not it receives a carry from the right. These transformations themselves happen to be quite structured, and they produce a lot of structure themselves, a feature that is unique to $(3n+1)$.
- The identities of the cells change after each transition, and the binary representation of the new number has to be parsed again against the described regular expression to identify the new cell types comprising the new number.
- However, one can still view the partitioning of the binary representation into cells that change their state based on their current state and information from adjacent cells as a form of cellular automata, and as we will see, the rules governing state transitions of such cells can directly be derived from the way the $(3n+1)/2$ rule, and its underlying shift and add operations work on binary strings.
- While in the rest of this presentation I won't use much math or number theory to formally prove properties about Collatz trajectories, still there is a one to one correspondence between a mathematical and number theoretic analysis of the changes that happen to a number as the Collatz rules are applied to it, and the changes that happen to its color-coded cellularised representation used in this presentation, and as such, the observable patterns in the trajectory of the representations, and any comprehensive analysis of them, one can argue can directly be used to make general claims about why the conjecture must be true, as the visualization can help us better see the underlying structure of this problem, and what are the driving forces as if inevitably pushing all numbers to descend to 1 when following the rules of $(3n+1)/2$.

Cell Transitions



The following table shows the major cell transitions for the $3n+1$ rule applied to odd numbers. Each entry in the table shows the before and after states of a certain configuration of a cell. The first row in any table entry shows the cell in the middle, with any carry input in a parenthesis to its right, and the rightmost bit of its left neighbor, if there is any. The second row shows how the bits in the cell change, and whether or not it generates a carry for its left neighbor. Note that the rightmost cell in the binary representation always receives an input carry. The empty cell to the left of the leftmost cell in the binary representation of a number may receive a carry input from it and end up with a 1 bit.






	1	2	3	4	5
A	<div>0 111111 (1)</div> <div>(1) 011111</div>	<div>1 000000 (0)</div> <div>(0) 100000</div>	<div>0 101010 (1)</div> <div>(1) 000000</div>	<div>1 101010 (1)</div> <div>(1) 100000</div>	<div>1 101010 (0)</div> <div>(1) 011111</div>
B	<div>0 111111 (0)</div> <div>(1) 011110</div>	<div>1 000000 (1)</div> <div>(0) 100001</div>	<div>0 101010 (0)</div> <div>(0) 111111</div>	<div>1 10 (1)</div> <div>(1) 10</div>	<div>1 10 (0)</div> <div>(1) 01</div>
C	<div>0 11 (0)</div> <div>(1) 00</div>	<div>1 00 (1)</div> <div>(0) 11</div>			
D	<div>0 1 (0)</div> <div>(0) 1</div>	<div>..... 0 (1)</div> <div>..... 1</div>			

Cell Transitions (2)

- It might seem that because of carry propagation, the cells need to wait for information to propagate to them before they can determine how they will have to change. However, it will be shown that given the type and other characteristics of a given cell and its neighbors (e.g., their size and whether or not the cell is at the LSB or MSB side), each cell can fully determine its own transformation.
- This observation about the cells suggest that they act like a type of cellular automata, and their computations can be all done locally and in parallel. However, after each application of the $3n+1$ rule, a re-segmentation of the binary representation is required, which makes this model different than traditional cellular automata where the cells persist.

Notable Patterns

- By construction (applying the segmentation rules from right to left), we never have a blue cell on the right side of a green cell. That is, a pattern like  is impossible, and instead would be encoded as .
- A blue cell never generates a carry out (see entries A2, B2 in the transition table). Thus a blue cell will flip the rightmost bit of the red cell to its left (B1), unless that cell is a single-bit red cell (D1) which is explained in a next bullet. Thus blue cells can pray on large red cells to their left. In the case of a single-bit red cell sandwiched between two blue cells, in the next cycle the red cell expands into two bits, however a 2-bit red cell can completely disappear under B1, and example of which is the right example below. Such red cells are transient and under the right conditions disappear.

	 (1)
	 (1)
	

- A single-bit red cell with a blue cell on its right and a green cell on its left will turn the green cell to all red, except for the leftmost 1 of the green cell, whose color will depend on the color of the original cell to the left of the green cell (see A5 and B3). Note that this is made possible by two rules. First, the blue cell does not generate any carries to the red cell, thus the red cell itself does not generate any carry to the green cell, and that's what makes A5 or B3 possible. This pattern is the *only* mechanism underlying spontaneous creation of large red cells. This can be proved by examining the entries of the cell transition table.

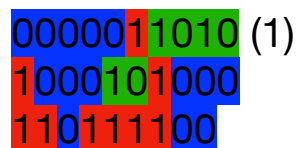



Notable Patterns (2)

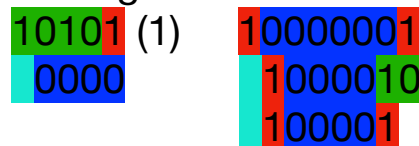
- Meanwhile, any red cell (large or single-bit) on the left side of any green cell (A4/A5/B4/B5) will always receive a carry, and as result will shrink in size (A1), and if it was a single-bit red cell, it will turn into a blue cell. Moreover, the original red cell will always send a carry out to any green cells on its left (A1), turning them into mostly blue (A3, and A4). A single-bit red cell sandwiched between two green cells can thus create the following two patterns, depending on whether or not the right green cell received a carry input or not (A4, A5). the case of A4 is shown below on the left. However, note that for A5 to apply, the right green should not be receiving any carries itself, and thus as described in the previous bullet above (add bulls numbers****), it must be on the left of a red-blue combination. This means that interleaved stretches of single-bit red cells and green cells generate interleaved stretches of red and blue cells (as shown in the rightmost example below), and as described in a previous bullet above (add bulls numbers****), such patterns preserve and contain the single-bit red cells.


 101011010 (1) 1010110101000 10101101011010 (1)
 00001000 000001111100 0000100001000

- Consider the case of a single-red cell sandwiched between a blue cell on the left, and a green cell on the right as shown below. This pattern in turn reduces to the green, single-bit-red, blue pattern in the next cycle, which was previously described.


 0000011010 (1)
 1000101000
 110111100

- Finally, a single red cell on the LSB end of the binary pattern reduces to a single blue cell according to A1, and is then eliminated by the $n/2$ rule. The cell transforms any green cell to its left to a mostly blue cell, which will also be eliminated by applications of the $n/2$ rule. When on the right side a blue cell, a rightmost red cell flips only the rightmost bit the blue cell, leaving behind another single-red cell, that might still be adjacent to a blue cell and in that case the pattern repeats.

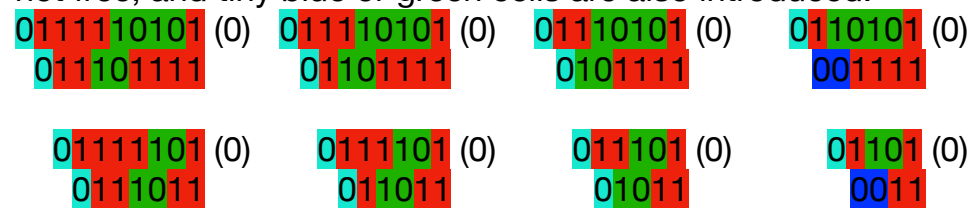

 10101 (1) 10000001
 0000 1000010
 100001

Notable Patterns (3)

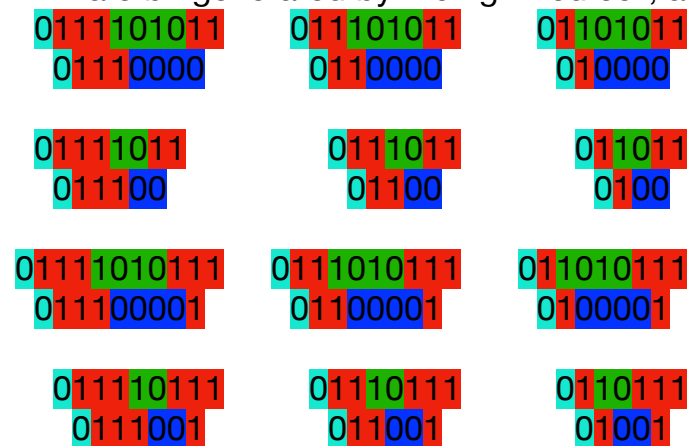
- As can be seen, single-bit red cells are what transforms green cells to large red cells under very specific conditions. Large red cells (ALL_ONES) on the right end of a bit pattern are what causes the bit pattern to get elongated, as they lead to repeated application of the $3n+1$ rule until all of their 1 bits are exhausted.
- Meanwhile, single-bit red cells also transform green cells into large blue cells, under a different, but larger set of conditions. Blue (ALL_ZEROS) cells on the right side of the bit pattern are what causes the bit pattern to get shortened (faster than ALT_10 green cells). Because the set of conditions under which blue cells get generated, or proliferated by, single-bit red cells is larger than the set of conditions under which red cells are generated by single-bit red cells, the blue cells seem to have more opportunities to develop.
- Moreover, since larger red cells always generate a carry, they always turn a green cell to their left to a blue cell (A3, A4).
- This makes the set of conditions under which red cells can transform a green cell to their right to a blue cell, versus a red cell, even bigger.
- However, one must also take into account the conditions under which green cells are generated, and whether those conditions favor the generation of red cells, and thus compensate, or even overwhelm, the balance towards creation of more red cells. These conditions are discussed shortly.
- In the subsequent sections I will show that blue cells still have other advantages over red cells that effectively tilts the balance of the opposite forces that shrink and expand the bit sequence towards more shrinkage, which leads to the creation of an overall attraction towards ultimate reduction of the number to 1.

Red-Green-Red

- Consider a red-green-red scenario where the right red cell is single-bit and does not receive a carry, and thus does not send a carry to the green cell on its left. The middle green cell always changes to a red/1 cell, except for its leftmost bit (A1, A5, D1). Note that the introduction of large red cells is not free, and tiny blue or green cells are also introduced.



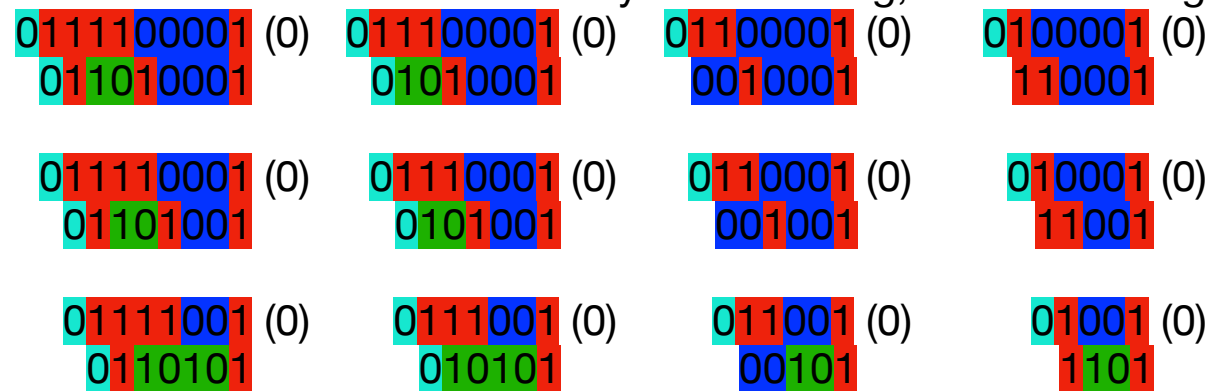
- Note that a single-bit red cell that does not receive a carry (D1) cannot possibly have a green cell to its right, because green cells on the right side of a red cell always send the red cell a carry (A4/A5). So, in the above scenarios, the rightmost single-bit red cell always has a blue cell to its right. Moreover, such single-bit red cell cannot possibly be the LSB of the number, because such LSB always receives a carry, and here we are considering only single-bit red cells not receiving a carry.
- Consider a red-green-red scenario where the right red cell is multi-bit that may or may not be receiving a carry (A1/B1), or it is a single-bit red cell that is the LSB of the number and thus always receives a carry (A1). In these two cases, the right red cell always sends a carry to the green cell. The middle green cell always changes to a blue/0 cell, except for its leftmost bit (A1, A4, A1/B1). Note how the 0 bits generated by the green cell merge with a 0 bit generated by the right red cell, as if encroaching on the right red cell.



- Notice how in all cases above, the middle green field, now turned blue or green, is also shifted to the right.

Red-Blue-Red (1)

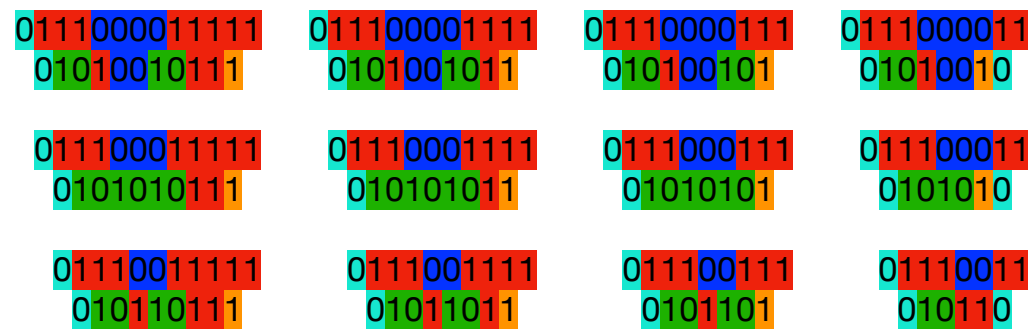
- Consider a red-blue-red scenario where the right red cell is single-bit and does not receive a carry, and thus does not send a carry to the blue cell on its left. The middle green cell always changes to a blue/0 cell, except for a 1/red MSB bit (B1, A2, D1). Note that if the size of the blue cell is 2, it effectively changes to a green cell. Meanwhile, the minimum size of a blue cell sandwiched between two red cells is 2. In the examples below, different sizes for the blue and left red cell are shown. The main emphasis is however on what happens to the middle blue cell, and not on all the things that can happen to the left red cell, as that would depend on the length of the left red cell and its left neighbor. Notice that if the middle blue cell is only two bits long, it turns into a green cell.



- Once again note that by construction, the single-bit red cell on the right side of the above examples must always be on the left side of blue cell.
- In all above scenarios, the right single-bit red cell is unaffected. If the left red cell is single-bit, it will be unaffected as well. Otherwise, its right two bits turns green. That is, the middle blue cell encroaches on the right edge of the left red cell if the red cell is multi-bit. If the middle blue cell is two bits only (e.g., the bottom of a blue triangle in the trajectory of a number), it turns into green, that can merge with any green cell produced by the left red cell. Such green cells resolve to (mostly) red or blue in the subsequent application of $(3n+1)/2$ rule, giving rise to the top of a triangle.

Red-Blue-Red (2)

- Consider a red-blue-red scenario where the right red cell is multi-bit that may or may not be receiving a carry (A1/B1), or it is a single-bit red cell that is the LSB of the number and thus always receives a carry (A1). In these two cases, the right red cell always sends a carry to the blue cell. The MSB and LSB bits of the middle blue cell become 1 (B1, B2, A1/B1). In the examples below, different sizes for the blue and right red cell are shown. The main emphasis is however on what happens to the middle blue cell, and not on all the things that can happen to the right red cell, as that would depend on the length of the right red cell and its right neighbor. The size of the red left cell is kept fixed, as the variations that it give rise to are covered in the red-blue-red examples of previous slide (with a single-bit red cell on the right that receives no carry).



- In the above examples, a red bit on the right end is colored with orange if its true class could be either red or green depending on what kind of cell is (blue or green) is to the right of the right red cell. Similarly, a 0 bit is colored cyan, if its true class could be either green or blue depending on what kind of cell (green or blue) is on the other side of the corresponding original red cell. Bits whose values have even more dependencies are omitted.
- In the above scenarios, what happens to the left red cell is similar to the scenarios where the right red cell is single-bit and sending no carry to the blue cell (discussed in the previous slide). However, this time the bits at the interface of the middle blue cell and the right multi-bit red cell change on both side of that interface. That is, the blue cell shrinks on its right edge as well as on its left edge, and the right red cell shrinks on its left edge.
- Note that in this scenarios, when the middle blue cell is three bits long, large green cells can form, from the remains of the three original cells, which with the subsequent application of $(3n+1)/2$ will create a large red or blue cell, which would in turn become the top of a large triangle in the rendered trajectory of the number.
- When the length of the middle green cell is not three, this scenario still produces green cells on the two sides of the original blue cell, that depending on the length of the original right and red cells, these green cells may become part of larger green cells.

RGR to RBR Cycle

- An interesting pattern that frequently occurs in a Red-Blue-Red pattern with only that 2 bits of 0 (i.e., 10011) that turns into Red-Green-Red (11101), and then changes back to (10011). The individual transitions have been presented in the Red-Blue-Red and Red-Green-Red analysis. The transitions are individually shown below:



01001111	01110111
11101111	01100111

- Below the transitions are lined up after each other. The 2-bit alternating 00/10 bit shifts to the right as long as there is a multi-bit red cell on its right. If the red cell is on the right edge of the number, when its bits are exhausted (corresponding to consecutive applications of $(3n+1)/2$), depending on whether the 2-bit field to its left is green (10) or blue (00) at that moment, a division by 2 or 4 becomes possible (corresponding to one or two applications of the $n/2$ rule).





01001111
11101111
01100111
11101

Blue-Green-Red

- Consider a blue-green-red where the right red cell is single-bit that does not receive a carry. Such a single-bit red cell does not send a carry to the green cell (D1). The middle green cell always changes to a red/1 cell (A2, B3, D1). The blue cell shrinks, but only on the left. This scenario is similar to red-green-red scenario (with single-bit red cell on the right) previously discussed, turning green cells to mostly red.

 (0)
  (0)

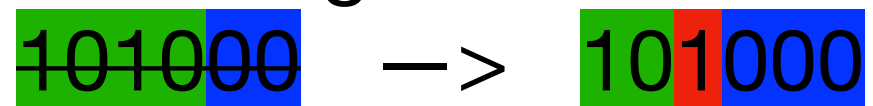
- Note that there is only one possibility for the bit to the left of the blue field in this scenario: it can only be red, by the way the binary string is parsed from right to left, always looking for maximal stretches of 1's, or 0's, and if not alternating 10's.
- Also note that the left blue cell has to be at least two bits long. The reason is that if it was a single bit, it would be absorbed to the green cell to its right together with the first red bit on its left.
- Also, note that the for the right red-cell to be single bit and not receiving any carry (D1), it has to be on the left side of a blue cell with at least two bits, because a green field on its right would always send it a carry (A4/A5).
- Consider a blue-red-green scenario where the right red cell is multi-bit that may or may not be receiving a carry (A1/B1), or it is a single-bit red cell that is the LSB of the number and thus always receives a carry (A1). The middle green cell always changes to a blue/0 cell (B2, A3, A1/B1). The blue cell shrinks by one bit on each size. Note how the 0 bits generated by the green cell merge with a 0 bit generated by the right red cell, as if encroaching on the right red cell.

 (1) LSB
  (1) LSB
  (1) LSB
  (1) LSB

- Notice how in all cases above, the middle green field, now turned blue or red, gets extended on its right, while maintaining its left edge.
- Finally, note that what happens to the middle green cell can be determined by the size and location of the right red cell; that is, whether the red cell is multi-bit or single-bit, and if the red cell is single-bit whether or not it is the LSB bit of the number. Assuming that cells could communicate their attributes (location and size) to their neighbors, such that the middle green cell knows the attributes of the right red cell, then the green cell can determine how its bits change when a $(3n+1)/2$ rule is applied.

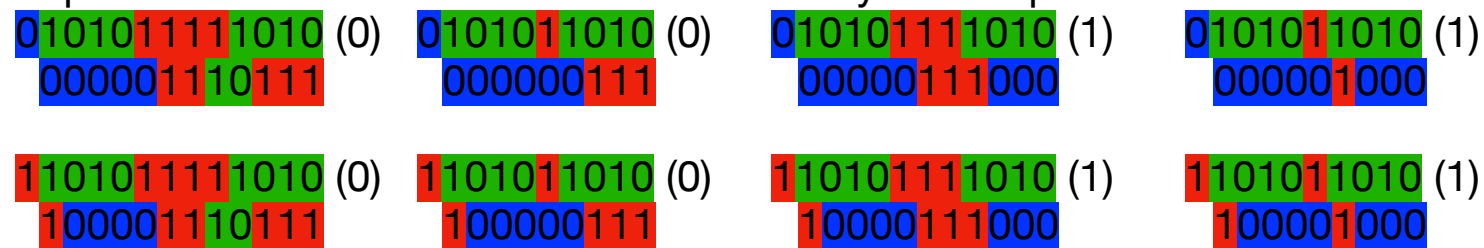
Green-Blue

- This combination will never happen, because of the way the binary string is parsed from right to left. The rightmost 0 in the green cell will always be counted as belonging to the blue cell, and the bit 1 to its left will be identified as a single-bit red cell. This is because when parsing a binary string from right to left, we prioritize finding maximal substrings of 0's or 1's, over 10's strings.

 101000 → 101000

Green-Red-Green

- Consider a green-red-green scenario. A green cell on the right side of a red cell always sends the red cell a carry (A4/A5). The red cell having received a carry loses its leftmost bit (A1), and always sends a carry to the left green cell (A3/A4), changing all but potentially the leftmost bit of the left green cell to 0/blue. The next state of the right green cell depends on whether or not it receives a carry. All the possibilities are shown for a 3 bit red cell.

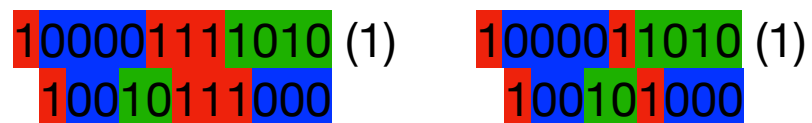


- Note that in all the above cases, the left green cell becomes (mostly) blue and also encroaches on the middle red cell.
- If the right green cell is receiving a carry, the middle red cell shifts to the right, but it also finds itself next to a blue cell to its right. However, if the right green cell does not receive a carry (i.e., if it is next to a single-bit red cell not receiving a carry, covered in a previous red-green-red scenario), then we would also have red cells generated from the right green cells, separated from the now shrunk red cell by a new green cell, unless the original red cell was single-bit, in which case it completely goes away, leading to a large blue cell.
- Note that what happens to the bits of the middle red cell is independent of whether or not the green cell receives a carry, because a green cell on the right side of a red cell always sends it a carry (A4/A5). Thus, the middle red cell can fully determine how to update its bits for a $(3n+1)/2$ step, solely by knowing that it is in a green-red-green configuration.
- Overall, there seem to be more opportunities for blue cells to proliferate than for red cells, but only if all other things can be considered equally likely.

Blue-Red-Green

- Consider a blue-red-green scenario. A green cell on the right side of a red cell always sends the red cell a carry (A4/A5). The red cell having received a carry loses its leftmost bit (A1), and always sends a carry to the left blue cell (B2). The left and right bits of the blue cell turn to red/1. The next state of the right green cell depends on whether or not it receives a carry. All the possibilities are shown for a 3 bit red cell.

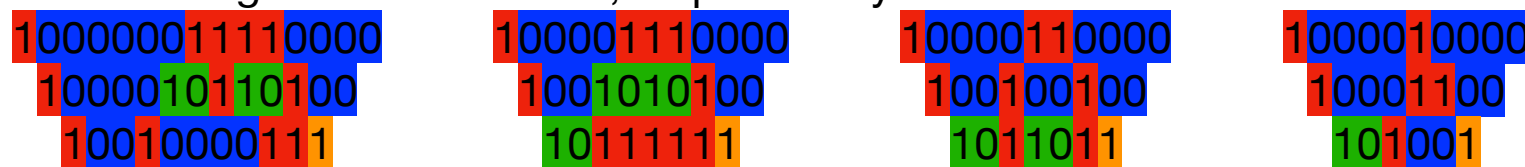

 1000011111010 (0) 1000011010 (0)
 100101110111 100100111


 100001111010 (1) 1000011010 (1)
 10010111000 100101000

- Note that if the middle red cell is multi-bit, the left blue cell shrinks on both sides, and a green cell gets generated on its right and to the left of the original red cell that shrinks on the left. If the middle red cell is single-bit, it disappears and becomes part of a blue or green cell.
- What happens to the red and green cells in this scenario is exactly like what happens to the middle and right cells in the green-red-green scenario previously discussed. As for what happens to the blue cell, it matches what was previously discussed for the red-blue-red scenario of the second type.
- Like the green-red-green case, what happens to the bits of the middle red cell is independent of whether or not the green cell receives a carry, because a green cell on the right side of a red cell always sends it a carry (A4/A5).

Blue-Red-Blue

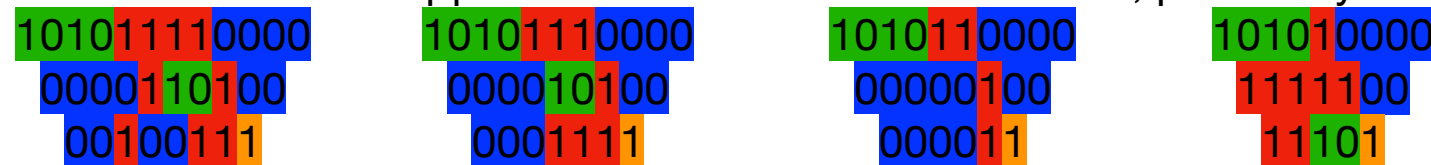
- Consider the blue-red-blue scenario. Note that by construction, both blue cells would be multi-bit (have two or more 0's). A blue cell never sends out a carry to the cell on its left (A2/B2). Thus the middle red cell in this scenario is not receiving a carry (B1). If the middle red cell is single-bit, it will persist and it won't send any carry out to the left blue cell (D1). But if the middle red cell is multi-bit it will lose a bit on either side of the cell (B1). If the middle red cell has two bits, they both become 0, becoming (part of a) blue cell (C1). The left blue cell shrinks on the left, but whether or not it shrinks on the right side as well depends on whether the middle red cell is multi or single-bit (B2/A1). The left and right bits of the blue cell turn to red/1.
- These scenarios can generate one or more green or blue cells when the size of the red cell is rather small. These green cells can be part of a red-blue-green, blue-green-red, green-red-green, red-green-red, green-red-blue, which have been covered elsewhere. Note the formation of a single-bit red cell on the left of a blue cell in many of the example shown (D1). When such single-bit red cell is the right red cell in a blue-green-red, or red-green-red scenario (as previous discussed), the green cell is transformed to (mostly) red. However, green-red-green formations generate blue cells, as previously discussed.



- Note that while a middle multi-bit red cell can produce many different patterns, the two blue cells adjacent to it shrink at their interface with the red cell. If the middle red cell is single-bit, the left blue cell does not shrink at its interface with the red cell. In this latter case, the red cell turns to blue with the subsequent application of $(3n+1)/2$.
- What happens to the bits of the middle red cell is independent of whether or not the right blue cell receives a carry, because a blue cell on the right side of a red cell (A2/B1) never sends the red cell a carry (B1).

Green-Red-Blue

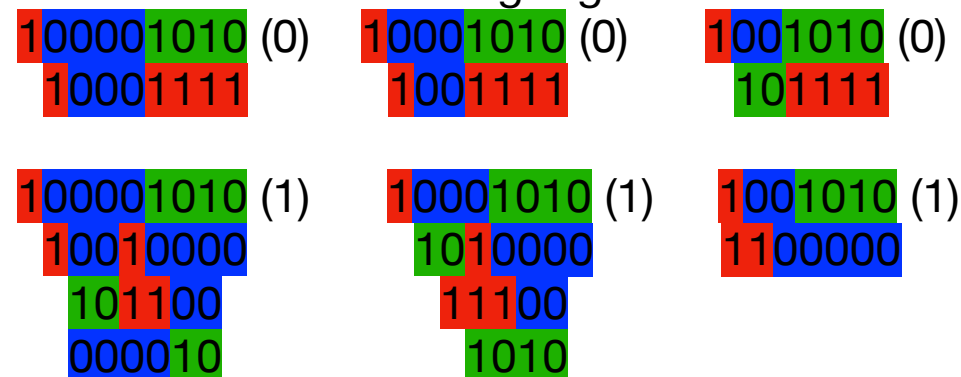
- Consider the green-red-blue scenario. Once again note that by construction, the blue cells would be multi-bit (have two or more 0's) and will never send out a carry to the red cell on its left (A2/B2). Thus the middle red cell in this scenario is not receiving a carry (B1 or D1). If the middle red cell is single-bit, it will persist and it won't send any carry out to the left green cell (D1). But if the middle red cell is multi-bit it will lose a bit on either side of the cell (B1). If the middle red cell has two bits, they both become 0, becoming (part of a) blue cell (C1).
- What happens to the left green cell not only depends on the size of the middle red cell (and thus whether or not the red cell sends it a carry), but also on what is on the left of the green cell. The many ways the green cell may change are already discussed in the context of red-green-red and blue-green-red scenarios. Thus here we choose to only focus on the middle red cell. However, what happens to the middle red cell and the blue cell to its right are the same as what happens in the blue-red-blue scenario, previously discussed.



- Note the formation of blue cells on the left, when the original middle red cell is multi-bit, versus formation of red cells when the middle red cell is single-bit. This is a phenomena we have noticed multiple times in the previous discussions. This difference in action is precisely due to the fact that the right blue cell does not send a carry to the red cell, and thus the affect of the red cell on the green cell is determined by the size of the red cell.
- There seems to be more opportunities for blue cells to form.
- What happens to the bits of the middle red cell is independent of whether or not the right blue cell receives a carry, because a blue cell on the right side of a red cell (A2/B1) never sends the red cell a carry (B1).

Red-Blue-Green (1)

- Note that a blue cell never sends out a carry to any cell on its left (A2/B2). Thus in a red-blue-green scenario the left red cell is either a single-bit red cell not receiving a carry (D1), or it is a multi-bit red cell not receiving a carry (B1).
- Meanwhile, the red-blue-green scenario is tightly related to the blue-green-red scenario, previously discussed. The reason is that the only type of cell possible on the left of a blue-green-red combination is a red cell, as previously explained when discussing the blue-green-red scenario.
- Also note that the middle blue cell has to be at least two bits long. The reason is that if it was a single bit, it would be absorbed to the green cell to its right together with the first red bit on its left.
- Thus, if we focus on what happens to the middle blue cell in a red-blue-green scenario, it is already covered by the blue-green-red scenario previously discussed. However, we can still look at some examples for this scenario, borrowed from blue-green-red examples whose rightmost red cell is removed. The results depend on whether or not the right green cell is receiving a carry.







Red-Blue-Green (2)

- Notice that in the case where the right green cell is not receiving any carry and thus not sending any carry to the middle blue cell (B3) is possible only if the right green cell is to the left of a single-bit red cell that has not received any carry (D1), which is only possible if such single-bit-red cell itself is to the left of a multi-bit blue cell. By construction, the cell to the right of the green cell can never be a blue cell.
- Notice the formation of spurious red cells in the bottom left two cases above (with a middle blue cell of size greater than 2, and the right green cell receiving a carry) that appear and then disappear. These spurious cells first form as a single-bit red cell in a blue-red-blue, or green-red-blue formation.
- Notice that in this scenario, the green cell is effectively passing through the carry that it receives to the blue cell, and how the bits of the blue cell change depend on the value of such carry. Thus, the middle blue cell cannot determine how to update its bits during a $(3n+1)/2$ step, solely based on its knowledge that it is in a red-blue-green configuration; it has to wait for the green cell to update itself and decide whether or not it is sending a carry to the blue cell. Alternatively, the middle blue cell has to know what type of red cell is on the right side of the green cell, and even what type of cell is to the right of such red cell. As can be seen, the blue cell cannot determine its next state only based on local information about its neighbors, and needs to reach out and have knowledge about farther away cells to its right. This issue only affects the question of whether or not the $(3n+1)/2$ step can be parallelized, such that each cell can locally update its bits using knowledge about its immediate neighbors, or perhaps a wider radius of neighbors. If not, the update rule has to be applied from the right end of the string of cells (representing the binary number) to the left, in a serial fashion, much like how carry needs to propagate from the LSB to MSB when performing any (i.e., binary) addition.

Green-Red LSB

- Consider a green-red scenario at the LSB of the number. Whether the right red cell is single-bit or multi-bit, it always receives a carry when applying the $3n+1$ rule (A1). Thus the green cell always changes to a blue/0 cell, except for its left-most bit that may become red/1, depending on whether the bit to the left of the green cell is 0 or 1 (A3 or A4). This scenario is thus similar to those blue-green-red or red-green-red scenarios previously discussed where the right red cell always sends a carry to the green cell.

0  (1) LSB
 0  (1) LSB
 1  (1) LSB
 1  (1) LSB





- Note how the 0 bits generated by the green cell merge with a 0 bit generated by the right red cell, as if encroaching on the right red cell.
- Finally, note that what happens to the green cell can be determined by the size and location of the right red cell; that is, whether the red cell is multi-bit or single-bit, and if the red cell is single-bit whether or not it is the LSB bit of the number. Assuming that cells could communicate their attributes (location and size) to their neighbors, such that the middle green cell knows the attributes of the right red cell, then the green cell can determine how its bits change when a $(3n+1)/2$ rule is applied.

Blue-Red LSB

- Consider a blue-red scenario at the LSB of the number. Whether the right red cell is single-bit or multi-bit, it always receives a carry when applying the $3n+1$ rule (A1). Thus the right red cell always sends a carry to the blue cell. The MSB and LSB bits of the middle blue cell become 1 (B1, B2, A1/B1). For further details, please refer to the section discussing red-blue-red scenarios (2) where the right red cell sends a carry to the blue cell. Notice that as explained before, a green-blue-red scenario is not possible, because by construction a green-blue scenario is not possible.

Green or Blue at LSB

- When a green or blue cell appears at the LSB of the number, the $n/2$ rule applies, which would remove all the LSB 0 bits of the right most blue/green cell, without affecting any of the other cells in the number. If the right-most cell is blue, it loses only a single 0. An example is shown below:

1010 LSB
101

MSB Red-Green

- Consider the scenario of a red-green pair of cells on the MSB of the number. The green cell (A4, A5) will always send a carry out to the red cell (A1), and thus the MSB bit of the red cell will flip to 0, and the red cell will always generate a carry out, the effect of which is that a new bit is introduced at the MSB of the number.

MSB Red-Blue

- Consider the scenario of a red-blue pair of cells on the MSB of the number. The blue cell will never send a carry out to the red cell (B1). Both the LSB and MSB bits of the red cell will flip to 0, and the red cell will always generate a carry out, the effect of which is that a new bit is introduced at the MSB of the number.

MSB Green-Red

- Consider the scenario of a green-red pair of cells on the MSB of the number. If red cell is sending a carry to the green cell (A1/B1), the green cell will become a blue/0 cell (A3), and generate a carry out, which would introduce a new bit, 1, at the MSB of the number. But if the red cell is not sending a carry to the green cell (D1), then the green cell will become a red/1 cell (B3), without producing any carry out, and thus no new bit will be introduced at the MSB of the number.

**The following slides are
other random thoughts
and/or TODOs**

- Instability of two bit cells: 00, 11, 10, that originate at the bottom and sides of triangles. They immediately turn to either red or blue, depending on the type of red cell (never a blue cell) that is to their right. When a green cell is turned into a red cell by a single-bit red cell (next to a blue cell) on its right, the larger red cell that gets generated can then turn any other green cells to its left to blue. Many time this happens right after generation of the larger cell. And if the length of the generated blue cell is just two bits, it keeps toggling along the red cell, turning into green and blue. If the red cell is on the right edge of the number, the 2-bit toggling blue-green cell will reach the right edge after the red cell is exhausted and shorten the number (via the $n/2$ action).
- Thus the overall dynamics is that both red and blue cells deteriorate in their edges, where the bits flip, which leads to alternating bits, and thus green cells, and green cells always immediately turn into red or blue cells by the red cells on their right; the red cells then give rise to more red and blue cells, and the blue cells ride along red cells on their right, and limit their reach to the other side of the blue cell. And that there are more opportunities for blue cells to form and persist, which leads to the descent of the number to 1.
- State that the visual analysis and enumeration of the different scenarios is equivalent to a more mathematical and number theoretic approach to the problem, in terms of studying the smaller factors that comprise the larger number and how they change as the Collatz rules modify the number. State that perhaps this is even a better approach to study this and similar problems, especially that our brains are so good with visual patterns and analyzing them (i.e., look at how much of math is driven from 2D-3D geometry), compared to more abstract types of math, like number theory, that can get quite complex the larger the numbers get.
- Red cells on the left side of the number rate limiting the blues
- Blue cells on the right side of the number limit the size of the large cells that get created
- Describing what limits the distribution of the sizes of the triangles (their tops); measuring that distribution; if it has to do with instability of two-bit patterns that form at the bottom of triangles, and the probability of two bit greens all lining up together, or when/how they happen together.

- Explain how the green cells form at the bottom of red/blue triangles
- Explain the actions of larger red cells
- Explain blue patches riding along the left side of red patches, reaching the LSB side
- Explain blue patches constraining/putting pressure on the size of red patches forming on the LSB end
- Explain blue/green patches eating up the right edge of red patches, except for when a red patch is at the LSB end
- Show that you don't need propagation if you have enough information about the color and size of your neighbors, and that this is indeed an instance of Macro-Cellular-Automata
- **Say that the Blues are connected, their tails travel down. While reds are not like that!**

- Present the $M2^K + (2^K - 1) \longrightarrow M3^K + (3^K - 1)$
- Describe that if there's ever a number that does not reach 1, it has to be in S1, and in particular a $6N+1$ number.
WHY? I don't remember why I go that idea?
- Make a note that a green-red combination where the red cell is single-bit with a blue cell on its right is really a 1010101 pattern, if we were parsing $(01)^+$ as well. The problem with parsing $(01)^+$ is that they will steal a 0 bit from a blue cell on the left, or create a green cells there, if the blue cell was two-bits.

Turing Machine?

- I can see that I can describe a Transition function for a Turing Machine that would read the binary string of the given number and manipulate it until its length goes to 1 and it halts.
- On the tape the 0 and 1 bits of the binary number are written, and at both ends of the number there's a special symbol X.
- The Transition function has two modes, the forward mode, and the backward mode. In the forward mode, the machine starts at the LSB end of the binary string with a carry (indicated in the current state of the machine), it reads each symbol (1 or 0), and based on whether there was a carry in (according to a bit in the state) and what was the previous symbol (stored in a bit in the state of the machine), it stores the current symbol in its state, computes and stores a carry in its own state, overwrites the current symbol, and moves the tape to the left, until it reaches an X. Note that it may need to overwrite the symbol at that X, if there is a carry in.
- Having reached a symbol X and potentially overwritten it, the machine then starts traversing the string backwards, until it reaches the LSB side of it again.
- At the start of the forward mode, the machine erases any 0 symbols, replacing them with X, until it reaches the first 1 symbol. If after the first 1 symbol there is an X, the the machine halts, as we have reached 1.
- The question is whether the machine always halts, for any arbitrary starting tape.
- <https://demonstrations.wolfram.com/CollatzSequenceComputedByATuringMachine/#more>
- In 1970, Russian mathematician [Yuri Matiyasevich](#) showed that [Hilbert's Tenth Problem](#), posed in 1900 as a challenge to the next century of mathematicians, cannot be solved. Hilbert's challenge sought an algorithm which finds all solutions of a [Diophantine equation](#). A Diophantine equation is a more general case of [Fermat's Last Theorem](#); we seek the [integer roots](#) of a [polynomial](#) in any number of variables with integer coefficients. Since we have only one equation but n variables, infinitely many solutions exist (and are easy to find) in the [complex plane](#); however, the problem becomes impossible if solutions are constrained to integer values only. Matiyasevich showed this problem to be unsolvable by mapping a Diophantine equation to a [recursively enumerable set](#) and invoking Gödel's Incompleteness Theorem.¹

1. Descriptions of real machine programs using simpler abstract models are often much more complex than descriptions using Turing machines. For example, a Turing machine describing an algorithm may have a few hundred states, while the equivalent deterministic finite automaton (DFA) on a given real machine has quadrillions. This makes the DFA representation infeasible to analyze.