

Blueprint for Failure

How to Construct a Counterexample to the Collatz Conjecture

By Paul Stadfeld

For comments: mensanator@aol.com

Version 1.1 Jun 2, 2006

Abstract:

A proof of the Collatz Conjecture has so far been elusive. But it may easily be resolved by finding a counterexample. In this paper we study the $3n+C$ extensions of the Collatz Conjecture (most of which fail) to learn what a counterexample to $3n+1$ would look like if it exists.

The study is based on the novel approach of creating a data structure (Sequence Vector) based on relative Collatz sequences without regard to their values. Functions are then derived from the data structure to allow algebraic analysis of the Collatz sequences.

These functions (Hailstone and Crossover Point) are the key to understanding why $3n+1$ works and why $3n+5$ fails. This understanding will then permit the construction of counterexamples as opposed to simple brute force searching. Furthermore, we will find that counterexamples must meet stringent structural requirements (the Blueprint) that will tell us when a counterexample CANNOT be constructed, which has deep implications concerning proving the Collatz Conjecture.

We will also gain new insight on the relationship between the positive and negative domain with regard to $3n+1$, that they are not separate problems but are actually a single problem from the viewpoint of relative Collatz sequences. This, too, has deep implications, for certain proofs in one domain imply proof in the other which seems to have been overlooked up to now with startling consequences.

The Collatz Conjecture will NOT be proved or disproved in this paper. What will be shown are tools and insights that may further the search for an actual proof.

1. Introduction

The Collatz Conjecture^{1,2} states that for any positive integer n , a sequence created using the rules

- 1) $\text{successor}(n) \rightarrow n/2$ if n is even
- 2) $\text{successor}(n) \rightarrow 3*n + 1$ if n is odd

always reaches the loop cycle 4, 2, 1, 4, 2, 1, Studied extensively, the conjecture appears to be true, but no proof has been found. It would, however, be easy to prove false. A counterexample can be readily verified. The fact that no counterexample has been located does not necessarily mean one does not exist. Current thinking is a counterexample in the form of a non-trivial loop cycle would have hundreds of thousands¹ of elements and may very well be beyond computational power for any foreseeable future.

That is, if we simply do a brute force search. Large numbers are no barrier if we have some blueprint that tells us how to build them. For example, the 1st 6th Generation Type [1,2] Mersenne Hailstone (see **Appendix B**) has 53,328 decimal digits and could never be found by simple brute force. Yet, *ith*, *kth* Generation Type [1,2] Mersenne Hailstones have a simple blueprint:

$$a(i, k) = 2 \left(6 * \left((i - 1) * 9^{(k - 1)} + \frac{(9^{(k - 1)} - 1)}{2} + 1 \right) - 1 \right) - 1$$

So the smart way to find a counterexample is to discover a blueprint that will take us directly to a counterexample even if it's unimaginably huge.

2. The Blueprint

A blueprint has been found together with the tools to construct a counterexample. They are based on how one navigates through the Collatz Tree³. The system presented here has deep implications, not just for tree navigation, but for the ultimate truth of the entire conjecture. It all starts with a data structure designed to efficiently describe relative pathways in the Collatz tree.

2.1. Sequence Vector

To navigate from one node on the Collatz tree to any other, one follows a sequence of Rules defined by the Collatz Conjecture. Although proper Collatz sequences always move right and down on the tree, there is no mathematical reason why one can't also move up and left (provided the moves are legal) by inverting the standard Collatz Rules.

Thus, we have

```
Rule1: n/2
Rule2: n*3+1
InverseRule1: n*2
InverseRule2: (n-1)/3
```

A rule based data structure describing a Collatz sequence is called a Sequence Vector.

By definition, a Sequence Vector always begins with an odd number and always ends with at least one iteration of Rule1 (the terminating number may be even or odd).

For example, the sequence starting on 27 and ending on 31

```
27_82
 41_124
   62
    31
```

is produced by the proper sequence

```
[Rule2, Rule1, Rule2, Rule1, Rule1]
```

We can abbreviate this by simply counting how many consecutive iterations there are of the same rule.

```
[1*Rule2, 1*Rule1, 1*Rule2, 2*Rule1]
```

Since a proper Collatz sequence cannot have more than 1 consecutive iteration of Rule2, we can further abbreviate the sequence by assuming every block of Rule1 is separated by exactly one Rule2 and only showing the Rule1 blocks (this assumption is why the Sequence Vector is defined to start on an odd number and end with a Rule1).

```
[1*Rule1, 2*Rule1]
```

Now, since every item shown is a block of Rule1, we can further abbreviate by just showing the iteration count of each block.

```
[1, 2]
```

In the above example, the sequence from 27 to 31 is just a small portion of the complete sequence from 27 to its normal termination at 1. The full sequence vector would be

```
[1, 2, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 3, 1, 1, 2, 1,
 2, 1, 1, 1, 1, 1, 3, 1, 1, 1, 4, 2, 2, 4, 3, 1, 1, 5, 4]
```

Sequence Vectors can describe a full sequence from n to 1 or any sub-sequence.

Thus, every proper Collatz sequence can be described by a simple list of integers >0 (0 is not a valid Sequence Vector element since the implied Rule2 to the left of each element would result in two consecutive iterations of Rule2, which is not permitted).

Sequence Vectors are not value-centric. They describe the structure of the sequence without specifying values. The sequence from 27 to 31 is a

Type [1, 2] Sequence Vector

since the [1, 2] pattern occurs many times on the Collatz tree (it occurs seven times in the full sequence from 27 to 1):

```
27_82
  41_124
    62
    31
. . .

107_322
  161_484
    242
    121
. . .

91_274
  137_412
    206
    103
. . .

155_466
  233_700
    350
    175
. . .

395_1186
  593_1780
    890
    445
. . .

251_754
  377_1132
    566
    283_850
      425_1276
        638
        319
```

(the last two occurrences of the seven [1,2] patterns are consecutive)

If we were to list all the Type [1, 2] Sequence Vector starting points,

3, 11, 19, 27, 35, 43, 51, 59, 67, 75, 83, 91, . . .

the *i*th occurrence (for *i*: 0, 1, 2, 3, ...) starts at

$$i * 8 + 3$$

And if we were to list all the Type [1, 2] Sequence Vector ending points,

4, 13, 22, 31, 40, 49, 58, 67, 76, 85, 94, . . .

the i th occurrence (for $i: 0, 1, 2, 3, \dots$) ends at

$$i * 9 + 4$$

These will be derived from any arbitrary Sequence Vector by the Hailstone Function.

2.2. Hailstone Function

For a generic Collatz sequence such as

$$\begin{array}{c} g_e \\ d_c \\ b \\ a \end{array}$$

the Sequence Vector is [1, 2].

We find the Hailstone Function by playing the Sequence Vector backwards using the InverseRules:

$$b = a * 2$$

$$c = a * 2 * 2$$

$$d = \frac{a * 2 * 2 - 1}{3}$$

$$e = \left(\frac{a * 2 * 2 - 1}{3} \right) * 2$$

$$g = \frac{\left(\frac{a * 2 * 2 - 1}{3} \right) * 2 - 1}{3}$$

working out all the algebra, we get

$$g = \frac{8 * a - 5}{9}$$

Now we need to solve the equation by finding an integer $a > 0$ such that g is also an integer:

$$\text{for } a=1 \quad g = \frac{8 * 1 - 5}{9} = \frac{8 - 5}{9} = \frac{3}{9}$$

$$\text{for } a=2 \quad g = \frac{8 * 2 - 5}{9} = \frac{16 - 5}{9} = \frac{11}{9}$$

$$\text{for } a=3 \quad g = \frac{8 * 3 - 5}{9} = \frac{24 - 5}{9} = \frac{19}{9}$$

$$\text{for } a=4 \quad g = \frac{8 * 4 - 5}{9} = \frac{32 - 5}{9} = \frac{27}{9} = 3$$

So 4 is the Hailstone derived from Sequence Vector [1, 2] whose Seed is 3.

Check: 3_10
 5_16
 8
 4

The Hailstone Function

$$g = \frac{8 * a - 5}{9}$$

can also be written using powers of 2 and 3:

$$g = \frac{2^3 * a - (3^0 * 2^1 + 3^1 * 2^0)}{3^2}$$

Each element i of the Sequence Vector (sv) applies the successor function

$$succ(a) = \frac{a * 2^{sv[i]} - 1}{3}$$

so if the Hailstone Function of [1, 2] is

$$g = \frac{2^3 * a - (3^0 * 2^1 + 3^1 * 2^0)}{3^2}$$

then the Hailstone Function for [3, 1, 2] would be

$$g = \frac{2^{3+3} * a - (3^0 * 2^{1+3} + 3^1 * 2^{0+3} + 3^2 * 2^0)}{3^{2+1}}$$

If we collect all the numerator terms inside the parentheses and call the collection Z, then every Sequence Vector has a Hailstone Function of the form

$$g = \frac{X * a - Z}{Y}$$

There is a relatively simple algorithm to calculate the values of X,Y,Z from a Sequence Vector so that we don't have to do all the algebra:

Hailstone Function Parameters Algorithm¹²

given a Sequence Vector of n elements [s0, s1, s2, ... sn-1]
 example: for [1, 2, 3, 4] n=4

X = 2 raised to the power of the sum of the elements

$$X = 2^{(1+2+3+4)}$$

$$X = 1024$$

Y = 3 raised to the power of the count of the elements

$$Y = 3^4$$

$$Y = 81$$

Z is a bit tricky, Z is the sum of n terms, each of which is a power of 3 multiplied by a power of 2

- create a template of n terms

$$Z = 3^? * 2^? + 3^? * 2^? + 3^? * 2^? + 3^? * 2^?$$

- the exponents of the powers of three count from 0 to n-1 from left to right

$$Z = 3^0 * 2^? + 3^1 * 2^? + 3^2 * 2^? + 3^3 * 2^?$$

- the exponents of the powers of 2 starting from the left are
 - the sum of all the elements except the last 1 (1+2+3)
 - the sum of all the elements except the last 2 (1+2)
 - the sum of all the elements except the last 3 (1)
 - the sum of all the elements except the last 4 (0)

$$Z = 3^0 * 2^6 + 3^1 * 2^3 + 3^2 * 2^1 + 3^3 * 2^0$$

$$Z = 1 * 64 + 3 * 8 + 9 * 2 + 27 * 1$$

$$Z = 64 + 24 + 18 + 27$$

$$Z = 133$$

for [1, 2, 3, 4] we get the Hailstone Function

$$g = \frac{1024 * a - 133}{81}$$

for which a=1 and g=11

11_34
 17_52
 26
 13_40

20
 10
 5_16
 8
 4
 2
 1

Solving the Hailstone function gives the first of infinite integer solutions. And if we know the first one (a_0, g_0) , we can find any solution. We saw in Section 2.1 that for Sequence Vector $[1, 2]$, the i th seed is $i*8+3$ and the i th hailstone is $i*9+4$. We can generalize these from the Hailstone Function:

$$a_i = i*Y + a_0$$

$$g_i = i*X + g_0$$

To find (a_0, g_0) for any Sequence Vector, we can reformulate the general Hailstone Function

$$g_0 = \frac{X * a_0 - Z}{Y}$$

as a problem in Linear Congruence^{4,5}: if g is an integer, then $(X*a-Z)$ is divisible by Y , in other words

$$X * a_0 \equiv Z \pmod{Y}$$

Linear congruence problems can be solved by using the Extended Euclidean Algorithm^{6,7} to find the modular inverse of X and Y :

$$a_0 = (\text{invert}(X, Y) * Z) \pmod{Y}$$

Now simply plug a_0 into the Hailstone Function to get g_0 .

A linear congruence can be solved if and only if the Greatest Common Divisor^{10,11} of X and Y divides Z . With X being a power of 2 and Y a power of 3, $\text{GCD}(X, Y)$ is always 1 and will always divide Z . This means that

- every Linear Congruence derived from a Hailstone Function is solvable
- every Hailstone Function has a first solution (a_0)
- every Hailstone Function has an infinite number of solutions (a_i)
- every legal Sequence Vector appears on the Collatz Tree infinitely many times

And from the Hailstone Function we calculate the Crossover Point.

2.3. Crossover Point

From any legal Sequence Vector, there exists a Hailstone Function with parameters X, Y, Z . From the Hailstone Function Parameters, we can compute the Crossover Point.

First, note that the Hailstone Function is the equation of a straight line (with slope X/Y and intercept $-Z/Y$). Because X is a power of 2 and Y is a power of 3, the slope can never be 1, and

so it cannot be parallel to the identity line whose slope is exactly 1. It must, therefore, intercept the identity line. This interception is called the Crossover Point.

Second, at the intercept point, since this is on the identity line, the function of a must be equal to a , so that

$$g = \frac{X * a - Z}{Y}$$

becomes

$$a = \frac{X * a - Z}{Y}$$

Thus, we make the following derivation:

$$\begin{aligned} Y * a &= X * a - Z \\ Y * a + Z &= X * a \\ Z &= X * a - Y * a \\ Z &= a * (X - Y) \\ \frac{Z}{(X - Y)} &= a \end{aligned}$$

But a is the point of intersection, so it is the Crossover Point (CP), so

$$CP = \frac{Z}{(X - Y)}$$

Every Sequence Vector/Hailstone Function has a Crossover Point, but if and only if the Crossover Point is an integer, then the Sequence Vector is a loop cycle! Every finite Collatz sequence has a first number which we'll call the Seed. Every finite sequence has a last number which we'll call the Hailstone. A Sequence Vector describes a loop cycle if Seed=Hailstone (i.e., the sequence returns to its starting value). The Crossover Point is the rational number $Z/(X-Y)$. If that rational number does not reduce to an integer, the Sequence Vector cannot be a loop cycle since a Collatz sequence contains only integers (i.e., the place where Seed=Hailstone cannot occur in a Collatz sequence). If it **is** an integer, then from the fact that this integer is also on the line of the identity function, that point represents an integer where Seed=Hailstone and thus, implies that the Sequence Vector is a loop cycle.

For example, take the Sequence Vector [2]

$$\begin{array}{c} g_c \\ b \\ a \end{array}$$

The associated Hailstone Function is

$$g = \frac{4 * a - 1}{3}$$

so

$$CP = \frac{1}{(4-3)} = \frac{1}{1} = 1$$

which means Sequence Vector [2] is a loop cycle.

$$\begin{array}{c} 1_4 \\ 2 \\ 1 \end{array}$$

By convention, for those Sequence Vectors that form loop cycles, the odd number of smallest magnitude in the loop cycle is called the Loop Point, so $a=1$ is the Loop Point of Sequence Vector [2].

If the Collatz Conjecture is true, there is only one positive Loop Point (1) whose Sequence Vector is [2] and it forms the root of the Collatz Tree. Any Loop Point/Sequence Vector other than (1)/[2] would be a counterexample.

2.4. $3n+C$ Extensions

The Crossover Point becomes our blueprint for studying how the Collatz Conjecture fails. Unfortunately, the only example of a Loop Point we have (Sequence Vector [2]) is trivial. But by extending the conjecture to $3n+C$ (where C is any odd positive integer), we can find numerous counterexamples to study via the Crossover Point.

First of all, a quick review of various values of C reveals that:

- The conjecture appears to be true for every C that is a power of 3 (including $C=1$).
- The conjecture appears to be false for every C not a power of 3

(See Section 3 for proof of these points)

Second, we need to make the following adjustments:

- The Extended Hailstone Function

$$g = \frac{X * a - Z * C}{Y}$$

The Hailstone Function Parameters X, Y, Z are invariant under the extensions.

- The Extended Crossover Point

$$CP = \frac{Z * C}{(X - Y)}$$

From the blueprint (Crossover Point), we will be able to determine exactly why the $3n+C$ systems work and why they fail. Once we fully understand the failure mechanism, we may be able to extend it back to $3n+1$ to construct a counterexample.

3. The Essence of Failure

There are two ways the conjecture can fail:

- A sequence goes to infinity.
- A sequence ends in a loop cycle other than the trivial loop cycle whose Sequence Vector is [2].

The possibility of a sequence going to infinity is beyond the scope of this paper and will not be mentioned again. Our blueprint for construction of a counterexample applies to the second case. But note that we specify Sequence Vector [2] as the sequence termination instead of 1. Under the $3n+C$ extensions, the trivial loop cycle has an invariant Sequence Vector. But the actual value of the Loop Point does vary. It is C .

$$\begin{array}{l} C_{4C} \\ 2C \\ C \end{array}$$

A terminating value of 1 is a value-centric specification that applies only to $3n+1$.

In addition, the trivial loop whose Sequence Vector is [1,2] and whose root is C , the tree built from that root is identical in structure to the $3n+1$ tree:

$$\begin{array}{l} 6C \ 20C \ 64C \\ 3C_{10C} \ 32C \\ 5C_{16C} \\ 8C \\ 4C \\ 2C \\ C \end{array}$$

Note that the coefficients of C are the same as the nodes on the $3n+1$ tree. So on the trivial loop tree of any $3n+C$ system, only multiples of C are on this tree. For any $3n+C$ system $n=1$ is never a multiple of C , so $n=1$ is never found on the trivial tree. Thus, $n=1$ is always found on a non-trivial tree. This proves the conjecture is false for any C not a power of 3 (as observed in Section 2.4).

Under $3n+C$, any Sequence Vector other than [2] whose Extended Crossover Point is an integer is a loop cycle and makes the conjecture false for that value of C . An integer implies that the denominator of the Extended Crossover Point

$$CP = \frac{Z * C}{(X - Y)}$$

is a unit, either because

- it started out as a unit ($X-Y$ is 1 or -1)

- it became a unit by factor canceling (the combined factors of Z^*C cancelled all the factors of $X-Y$)

3.1. Trivial Loops

When $X-Y$ is a unit (as is the case for Sequence Vector [2]), CP is trivially an integer and forms a trivial loop cycle. Trivial loop cycles are determined solely by the Sequence Vector and are common to all $3n+C$ systems.

Thus, every $3n+C$ system has a trivial Loop Point at C . And no others since 4-3 is the only pair of powers of 2 and powers of 3 whose difference is +1.

3.2. Non-Trivial Loops

That leaves factor canceling as the only means of failure. And there's plenty of it.

Example $3n+5$:

trivial loop at 5:	5, 20, 10, 5
non-trivial loop at 1:	1, 8, 4, 2, 1
non-trivial loop at 19:	19, 62, 31, 98, 49, 152, 76, 38,19
non-trivial loop at 23:	23, 74, 37, 116, 58, 29, 92, 46, 23
non-trivial loop at 187:	187, 566, 283, 854, 427, 1286, 643, 1934, 967, 2906, 1453, 4364, 2182, 1091, 3278, 1639, 4922, 2461, 7388, 3694, 1847, 5546, 2773, 8324, 4162, 2081, 6248, 3124, 1562, 781, 2348, 1174, 587, 1766, 883, 2654, 1327, 3986, 1993, 5984, 2992, 1496, 748, 374, 187
non-trivial loop at 347:	347, 1046, 523, 1574, 787, 2366, 1183, 3554, 1777, 5336, 2668, 1334, 667, 2006, 1003, 3014, 1507, 4526, 2263, 6794, 3397, 10196, 5098, 2549, 7652, 3826, 1913, 5744, 2872, 1436, 718, 359, 1082, 541, 1628, 814, 407, 1226, 613, 1844, 922, 461, 1388, 694, 347

$3n+5$ has 5 non-trivial Loop Points: 1, 19, 23, 187 and 347. Picking one of these Loop Points, say 187, its Sequence Vector is

[1,1,1,1,1,2,1,1,2,1,2,3,2,1,1,1,5]

From which we get the Extended Hailstone Function

$$g = \frac{134217728 * a - 189900931 * 5}{129140163}$$

And Extended Crossover Point (C is kept separate in the numerator to illustrate factor canceling)

$$\begin{aligned}
CP &= \frac{189900931 * 5}{134217728 - 129140163} \\
&= \frac{189900931 * 5}{5077565} \\
&= \frac{(11 * 17 * 71 * 14303) * (5)}{5 * 71 * 14303}
\end{aligned}$$

When the common factors 5, 71 and 14303 are cancelled, we are left with 11*17=187 which is the Loop Point for this loop cycle. But since the Sequence Vector is a loop cycle, every cyclic permutation of the Sequence Vector must also have an integer Extended Crossover Point. And since in a cyclic permutation, the count and sum of the elements are invariant, only Z changes in the Extended Crossover Point.

[1,1,1,1,2,1,1,2,1,2,3,2,1,1,1,5,1]	(283*71*14303)*(5)/(5*71*14303)
[1,1,1,2,1,1,2,1,2,3,2,1,1,1,5,1,1]	(7*61*71*14303)*(5)/(5*71*14303)
[1,1,2,1,1,2,1,2,3,2,1,1,1,5,1,1,1]	(643*71*14303)*(5)/(5*71*14303)
[1,2,1,1,2,1,2,3,2,1,1,1,5,1,1,1,1]	(967*71*14303)*(5)/(5*71*14303)
[2,1,1,2,1,2,3,2,1,1,1,5,1,1,1,1,1]	(1453*71*14303)*(5)/(5*71*14303)
[1,1,2,1,2,3,2,1,1,1,5,1,1,1,1,1,2]	(1091*71*14303)*(5)/(5*71*14303)
[1,2,1,2,3,2,1,1,1,5,1,1,1,1,1,2,1]	(11*149*71*14303)*(5)/(5*71*14303)
[2,1,2,3,2,1,1,1,5,1,1,1,1,1,2,1,1]	(23*107*71*14303)*(5)/(5*71*14303)
[1,2,3,2,1,1,1,5,1,1,1,1,1,2,1,1,2]	(1847*71*14303)*(5)/(5*71*14303)
[2,3,2,1,1,1,5,1,1,1,1,1,2,1,1,2,1]	(47*59*71*14303)*(5)/(5*71*14303)
[3,2,1,1,1,5,1,1,1,1,1,2,1,1,2,1,2]	(2081*71*14303)*(5)/(5*71*14303)
[2,1,1,1,5,1,1,1,1,1,2,1,1,2,1,2,3]	(11*71*71*14303)*(5)/(5*71*14303)
[1,1,1,5,1,1,1,1,1,2,1,1,2,1,2,3,2]	(587*71*14303)*(5)/(5*71*14303)
[1,1,5,1,1,1,1,1,2,1,1,2,1,2,3,2,1]	(883*71*14303)*(5)/(5*71*14303)
[1,5,1,1,1,1,1,2,1,1,2,1,2,3,2,1,1]	(1327*71*14303)*(5)/(5*71*14303)
[5,1,1,1,1,1,2,1,1,2,1,2,3,2,1,1,1]	(1993*71*14303)*(5)/(5*71*14303)

Although Z changes for every Sequence Vector, note that every Z contains the factors 71 and 14303. Combined with the 5 factor from C, every denominator becomes 1 and every Extended Crossover Point becomes an integer. And if you check, you'll find that the 17 integer Extended Crossover Points are the 17 odd numbers in the original sequence. The convention of taking the smallest magnitude number of the loop cycle as the Loop Point is so that we don't have to constantly list all the cyclic permutations. When we refer to a Loop Point and its Sequence Vector, it is assumed that we are collectively referring to all the cyclic permutations unless otherwise noted.

Equally important, Z never has 5 as a factor (otherwise, this Sequence Vector would be a loop in $3n+1$ also). It is the combined factors of Z and C that create non-trivial loops. This immediately leads to the

Z Conjecture The factors of Z alone cannot cancel all the factors of the denominator, C must contribute at least one useful factor (where "useful" means >3).

A factor must be "useful" because the denominator, being the difference between a power of 2 and a power of 3 can never have a 3 as a factor. This explains why when C is a power of 3, it doesn't fail the conjecture. Or perhaps we should say it has exactly the same loop cycles as

$3n+1$ because 1 has no useful factors either. If the conjecture is true for $3n+1$, it's true for all C that are a power of 3 (as observed in section 2.4).

3.3. Factor Congruence

If we are given C , we can now construct Sequence Vectors that are non-trivial loops. This is done via Factor Congruence matching. For example, suppose C is a prime (41). Being prime, C has only one factor: 41. This factor is the only one available to cancel the factors uncanceled by Z . So for CP to be an integer, it is necessary that $X-Y$ has a factor of 41. In other words

$$X - Y \equiv 0 \pmod{41}$$

or

$$X \equiv Y \pmod{41}$$

From The Hailstone Function, we know that X is a power of 2 and Y is a power of 3. So that

$$2^p \equiv 3^q \pmod{41}$$

Also from the Hailstone Function, we have that

$$p = \text{sum}(sv) \quad q = \text{length}(sv)$$

where sv is the Sequence Vector. Thus, only those Sequence Vectors where a power of 2 is the same congruence class mod 41 as a power of 3 can be a potential loop (potential because Z must supply the rest of the factors).

With our example 41, the congruence classes are

$$2^p = \{1, 2, 4, 8, 16, 32, 23, 5, 10, 20, 40, 39, 37, 33, 25, 9, 18, 36, 31, 21\}$$

$$3^q = \{1, 3, 9, 27, 40, 38, 32, 14\}$$

The only classes they have in common are: $\{1, 9, 32, 40\}$

p or q	$2^p \pmod{41}$ (pattern repeats every 20)	$3^q \pmod{41}$ (pattern repeats every 8)
0	1	1
1	2	3
2	4	9
3	8	27
4	16	40
5	32	38
6	23	32
7	5	14
8	10	1
9	20	3
10	40	9
11	39	27
12	37	40

13	33	38
14	25	32
15	9	14
16	18	1
17	36	3
18	31	9
19	21	27
20	1	40
21	2	38
22	4	32
23	8	14

Table 1: Congruence classes of powers of 2 & 3 mod 41.

So p and q must be chosen so that the congruence classes match. The first such match is $p=0$ and $q=0$ match. But a Sequence Vector cannot have 0 length or have a 0 sum, so that pairing is disallowed. To be a legal Sequence Vector p must be $\geq q$. And to be a positive loop, it must be that $p > q * (\log 3 / \log 2)$. Under those restrictions, the first available pair is $pq(15,9)$, where both congruence classes are 9. The next available one is $pq(10,4)$ for congruence class 40. The pair $pq(5,6)$ cannot form a loop even though they are both congruence class 32 because p must be greater than q (in other words, there are no legal Sequence Vectors with more $3n+C$ steps than $n/2$ steps). And any loop in $pq(20,16)$ would be in the negative domain since $p < q * (\log 3 / \log 2)$. A good candidate would be $pq(20,8)$.

But there are many different Sequence Vectors where $p=20$ and $q=8$. Constructing them is equivalent to partitioning 20 objects into 8 containers such that each container has at least 1 object. Using an algorithm¹³ for counting and generating such partitions, we can search every partition for an integer Crossover Point. For (20,8) the number of partitions is 50,388 and we have to check them all because Factor Congruence matching is a necessary but not sufficient condition for loop cycle formation.

Running the Partition Generator for $p=20$ $q=8$ and testing for integer Crossover Points yields

```
[1,1,9,2,1,3,1,2] 47
[1,2,1,1,9,2,1,3] 19
[1,3,1,2,1,1,9,2] 11
[1,9,2,1,3,1,2,1] 91
[2,1,1,9,2,1,3,1] 49
[2,1,3,1,2,1,1,9] 1
[3,1,2,1,1,9,2,1] 37
[9,2,1,3,1,2,1,1] 157
```

So there are 8 Sequence Vectors that produce loops. But upon close inspection, we see that all 8 are cyclic permutations of the same numbers. That means the 8 integer Crossover Points are the 8 odd numbers of a single loop cycle:

```
1_44
 22
 11_74
 37_152
 76
 38
```

```

19_98
 49_188
   94
  47_182
   91_314
    157_512
     256
      128
       64
        32
         16
          8
           4
            2
             1

```

1 is a Loop Point in $3n+41$ whose Sequence Vector [2,1,3,1,2,1,1,9] is a non-trivial loop cycle.

QED

See *Appendix A* for an analysis of a composite C .

Using the Extended Crossover Point as our blueprint and Factor Congruence matching as our tools, we now have the means to construct counterexamples to the $3n+C$ extensions. And although the lack of useable factors prevents us from constructing any counterexample to $3n+1$, it would be premature to leap to the conclusion that $3n+1$ must be true. Up to this point, we have been assuming the *Z Conjecture* has been true. Actually, it doesn't matter to Factor Congruence matching whether the *Z Conjecture* is true or not. If it's false, the loop cycles constructed by Factor Congruence matching are still loop cycles, it's just that they won't be the only non-trivial loop cycles. For $3n+1$ to be true, we must show that the *Z Conjecture* is true so that when we say no loop cycles can be constructed by Factor Congruence matching, then no loop cycles can be constructed period.

For that we will need to make one more extension.

4. The Negative Domain

The Collatz Conjecture is specified as a problem in the positive domain. We can, of course, extend it to the negative domain just as we extended $3n+1$ to $3n+C$. We quickly discover that the negative domain is rather uninteresting in that the conjecture fails for every C in $3n+C$. Every $3n+C$ (including $3n+1$) has Loop Points at $-C$, $-5C$ and $-17C$.

But we're looking to understand why it fails, that's how we were able to derive the Blueprint for Failure in the preceding sections. And although there is nothing in the negative domain that teaches us anything about trivial loops or Factor Congruence that we don't already know, it does contain this bombshell:

Proof that the *Z Conjecture* is false!

The Loop Points at $-C$ and $-5C$ (with Sequence Vectors [1] and [1,2] respectively) are trivial because from their Crossover Points, $(X-Y)$ starts out as a unit (-1). There is no need for factor canceling.

But $-17C$ is **not** trivial and forms a loop cycle even when $C=1$:

Sequence Vector [1, 1, 1, 2, 1, 1, 4]

```

g_s
 r_q
  p_o
   n_m
    l
     k_j
      i_h
       f_e
        d
         c
          b
           a

```

$$g = \frac{2048*a - 2363}{2187}$$

$$CP = \frac{2363}{(2048 - 2187)} = \frac{2363}{-139} = \frac{17 * 139}{-139} = -17$$

```

-17_-50
 -25_-74
  -37_-110
   -55_-164
    -82
     -41_-122
      -61_-182
       -91_-272
        -136
         -68
          -34
           -17

```

The Crossover Point is an integer because the factors of Z cancel all the factors of $X-Y$ proving that the Z Conjecture is actually false. It is possible for all the factors of $X-Y$ to be cancelled without C contributing any useful factors. So even though the counterexample -17 is in the negative domain, the fact that it exists at all has deep implications for the positive domain. We still are unable to construct a counterexample to $3n+1$ via Factor Congruence matching, but we cannot rule out that a construction based on the Z Conjecture being false is possible.

5. Conclusions

As we have seen, a counterexample must be one of three types:

- Trivial loop cycles
- Factor Congruence loop cycles
- *Z Conjecture* loop cycles

The Trivial loops for $3n+C$ are $+C$, $-C$ and $-5C$. Factor Congruence loop cycles have been proven impossible to construct for $3n+1$. That leaves the *Z Conjecture* as the only way $3n+1$ can be false.

Unfortunately, we have no blueprint that can exploit the *Z Conjecture* being false, so we cannot construct counterexamples even knowing such a counterexample exists. And the fact that the only known counterexample to the *Z Conjecture* is in the negative domain doesn't mean there can't be one in the positive domain. Factor canceling is independent of the domain, so we can't say it's impossible for there to be a counterexample in the positive domain of $3n+1$.

Thus, the Collatz Conjecture continues to be as elusive as ever.

-o0o-

Appendix A

3n+85085, a Factor Congruence case study

From Section 3.3, we found the non-trivial Loop Point by searching only those Sequence Vectors whose Hailstone Function parameters had matching congruence classes mod C.

But not all matching parameters generated Loop Points. And what if C were composite, say 85085. Why 85085? Because it is the composite of the 5 primes > 3 : $85085 = 5 * 7 * 11 * 13 * 17$. Thus, there will be a lot of factors available to the Crossover Point Function to form Loop Points.

Running the collatz_C function¹⁴ for C=85085 over the range -2000000 to +2000000, a total of 156 Loop Points were located.

When C is composite, some of its factors may not be necessary to form a Loop Point. In such cases, the unused factors will appear as the greatest common divisor of the odd numbers in the loop sequence. So the gcd will tell us the divisor whose congruence classes must be matched:

divisor = 85085/gcd

And just as factor congruence matching is necessary but not sufficient for Loop Points, there can be any number of Loop Points found for any given divisor.

Since there are 5 factors in 85085 any of which may or may not be cancelled, there are 32 possible gcd/divisor pairs:

fac5	fac7	fac11	fac13	fac17	divisor	gcd
1	1	1	1	1	1	85085
1	1	1	1	17	17	5005
1	1	1	13	1	13	6545
1	1	1	13	17	221	385
1	1	11	1	1	11	7735
1	1	11	1	17	187	455
1	1	11	13	1	143	595
1	1	11	13	17	2431	35
1	7	1	1	1	7	12155
1	7	1	1	17	119	715
1	7	1	13	1	91	935
1	7	1	13	17	1547	55
1	7	11	1	1	77	1105
1	7	11	1	17	1309	65
1	7	11	13	1	1001	85
1	7	11	13	17	17017	5
5	1	1	1	1	5	17017
5	1	1	1	17	85	1001
5	1	1	13	1	65	1309
5	1	1	13	17	1105	77
5	1	11	1	1	55	1547
5	1	11	1	17	935	91
5	1	11	13	1	715	119
5	1	11	13	17	12155	7
5	7	1	1	1	35	2431
5	7	1	1	17	595	143
5	7	1	13	1	455	187
5	7	1	13	17	7735	11
5	7	11	1	1	385	221
5	7	11	1	17	6545	13
5	7	11	13	1	5005	17
5	7	11	13	17	85085	1

The 156 Loop Points are distributed such that all possible gcds are represented. So we have to calculate matches for each of the 32 divisors.

But we're just trying to verify that Factor Congruence matching actually works, so instead, we'll take the known Loop Points and see if they hold up to the congruence matching scrutiny.

See *Table A1*.

Which they do. That means we can verify the Loop Points with the partition generator program. Generating every partition of P items in Q groups is equivalent to creating every possible Sequence Vector whose sum is P and count is Q.

So, grabbing a P,Q pair from the stats table

```
python partition_generator.py 4 3
searching: 3 Sequence Vectors
```

```
[1,1,2] -146965      *
[1,2,1] -177905
[2,1,1] -224315
```

we get three Sequence Vectors with integer Crossover Points (the one marked * is the Loop Point shown on the stats table).

But note that the three vectors are all cyclic permutations of the same numbers. that means they are all part of the same loop:

```
-146965__-355810
      -177905__-448630
            -224315__-587860
                  -293930
                        -146965
```

So the partition generator will return exactly Q Sequence Vectors for each Loop Point found.

Exactly?

```
python partition_generator.py 4 2
searching: 3 Sequence Vectors
```

```
[1,3] 60775
[3,1] 133705
[2,2] 85085
```

Here Q is 2 but we got 3 vectors. But that's because the third vector is a different loop cycle. But it only has 1 vector. That's because it's actually a second generation type [2] vector, so we'll mark these false positives with an x. The multi-generation Loop Points will always have less than Q vectors.

```
python partition_generator.py 6 4
searching: 10 Sequence Vectors
```

```
[1,1,1,3] -325325      *
[1,1,3,1] -445445
[1,3,1,1] -625625
[3,1,1,1] -895895

[1,1,2,2] -365365      *
[1,2,2,1] -505505
[2,2,1,1] -715715
[2,1,1,2] -515515

[1,2,1,2] -425425      x
[2,1,2,1] -595595
```

Here the stats table shows two Loop Points for P=6 Q=4 and that's what we got, not counting the multi-generation vector.

More interesting is the set of Loop Points found for P=8 Q=5

```
python partition_generator.py 8 5
searching: 35 Sequence Vectors
```

```

[1,1,1,1,4] 1380995 *
[1,1,1,4,1] 2114035
[1,1,4,1,1] 3213595
[1,4,1,1,1] 4862935
[4,1,1,1,1] 7336945

[1,1,1,2,3] 1485715 *
[1,1,2,3,1] 2271115
[1,2,3,1,1] 3449215
[2,3,1,1,1] 5216365
[3,1,1,1,2] 3933545

[2,1,1,1,3] 2231845
[1,1,1,3,2] 1695155 *
[1,1,3,2,1] 2585275
[1,3,2,1,1] 3920455
[3,2,1,1,1] 5923225

[1,1,2,1,3] 1642795 *
[1,2,1,3,1] 2506735
[2,1,3,1,1] 3802645
[1,3,1,1,2] 2873255
[3,1,1,2,1] 4352425

[1,1,2,2,2] 1852235 *
[1,2,2,2,1] 2820895
[2,2,2,1,1] 4273885
[2,2,1,1,2] 3226685
[2,1,1,2,2] 2441285

[1,2,1,1,3] 1878415 *
[2,1,1,3,1] 2860165
[1,1,3,1,2] 2166395
[1,3,1,2,1] 3292135
[3,1,2,1,1] 4980745

[1,2,1,2,2] 2087855 *
[2,1,2,2,1] 3174325
[1,2,2,1,2] 2402015
[2,2,1,2,1] 3645565
[2,1,2,1,2] 2755445

```

Once separated into groups of cyclic permutations, we find we have seven groups - seven Loop Points. But the stats table only lists six. Look at the Loop Point on the last loop: 2087855.

The original Loop Point search only extended to 2000000, so the partition generator found one missed in the original search. The Sequence Vector search will always find all the Loop Points...

...provided we know what P and Q to test. Doubling the collatz_C test range, we find yet another Loop Point: 3182179.

This one didn't get picked up in the Sequence Vector search because its loop has P=27 Q=17 and that combination hadn't appeared before in the stats table. That warrants another run of the partition generator.

```
python partition_generator.py 27 17
searching: 5311735 Sequence Vectors
```

```

[1,1,1,1,1,2,1,1,2,1,2,3,2,1,1,1,5] 3182179 *
[1,1,1,1,2,1,1,2,1,2,3,2,1,1,1,5,1] 4815811
[1,1,1,2,1,1,2,1,2,1,2,3,2,1,1,1,5,1,1] 7266259
[1,1,2,1,1,2,1,2,1,2,3,2,1,1,1,5,1,1,1] 10941931
[1,2,1,1,2,1,2,1,2,3,2,1,1,1,5,1,1,1,1] 16455439
[2,1,1,2,1,2,1,2,3,2,1,1,1,5,1,1,1,1,1] 24725701
[1,1,2,1,2,3,2,1,1,1,5,1,1,1,1,1,1,2] 18565547
[1,2,1,2,3,2,1,1,1,5,1,1,1,1,1,1,2,1] 27890863
[2,1,2,3,2,1,1,1,5,1,1,1,1,1,2,1,1] 41878837
[1,2,3,2,1,1,1,5,1,1,1,1,1,2,1,1,2] 31430399
[2,3,2,1,1,1,5,1,1,1,1,1,2,1,1,2,1] 47188141
[3,2,1,1,1,5,1,1,1,1,1,2,1,1,2,1,2] 35412377
[2,1,1,1,5,1,1,1,1,1,2,1,1,2,1,2,3] 13290277
[1,1,1,5,1,1,1,1,1,2,1,1,2,1,2,3,2] 9988979
[1,1,5,1,1,1,1,1,2,1,1,2,1,2,3,2,1] 15026011

```

```

[1,5,1,1,1,1,1,2,1,1,2,1,2,3,2,1,1] 22581559
[5,1,1,1,1,1,1,2,1,1,2,1,2,3,2,1,1,1] 33914881

[1,2,1,2,2,1,1,1,1,3,1,1,1,1,2,2,4] 6109103
[2,1,2,2,1,1,1,1,1,3,1,1,1,1,2,2,4,1] 9206197
[1,2,2,1,1,1,1,1,3,1,1,1,1,2,2,4,1,2] 6925919
[2,2,1,1,1,1,1,3,1,1,1,1,2,2,4,1,2,1] 10431421
[2,1,1,1,1,1,3,1,1,1,1,1,2,2,4,1,2,1,2] 7844837
[1,1,1,1,3,1,1,1,1,1,2,2,4,1,2,1,2,2] 5904899 *
[1,1,1,3,1,1,1,1,1,2,2,4,1,2,1,2,2,1] 8899891
[1,1,3,1,1,1,1,2,2,4,1,2,1,2,2,1,1] 13392379
[1,3,1,1,1,1,1,2,2,4,1,2,1,2,2,1,1,1] 20131111
[3,1,1,1,1,2,2,4,1,2,1,2,2,1,1,1,1] 30239209
[1,1,1,1,2,2,4,1,2,1,2,2,1,1,1,1,3] 11350339
[1,1,1,2,2,4,1,2,1,2,2,1,1,1,1,3,1] 17068051
[1,1,2,2,4,1,2,1,2,2,1,1,1,1,3,1,1] 25644619
[1,2,2,4,1,2,1,2,2,1,1,1,1,3,1,1,1] 38509471
[2,2,4,1,2,1,2,2,1,1,1,1,3,1,1,1,1] 57806749
[2,4,1,2,1,2,2,1,1,1,1,3,1,1,1,1,2] 43376333
[4,1,2,1,2,2,1,1,1,1,3,1,1,1,1,2,2] 32553521

```

And, lo and behold, another Loop Point is found: 5904899.

Unfortunately, verification was halted before the end of the stats table was reached. For the simple reason that as P and Q get large, Sequence Vector searching becomes intractable:

```
python partition_generator.py 48 24
searching: 16123801841550 Sequence Vectors
```

```
python partition_generator.py 52 24
searching: 196793068630200 Sequence Vectors
```

```
python partition_generator.py 56 32
searching: 2488589544741300 Sequence Vectors
```

```
python partition_generator.py 60 30
searching: 59132290782430712 Sequence Vectors
```

```
.
.
.
```

```
python partition_generator.py 492 264
searching:
6618941526433155948277640969499303240702870967755059629130019289014193777349831417543311612293
9513634124491233746912456893016976209252459301489030
Sequence Vectors
```

And there may be some more hiding in other congruence match pairs that aren't on the original stats table. Noting that the congruence generations reach as high as 13, 4 there may be others.

And if there are, they'll fall into their proper place in the Factor Congruence table.

Table A1: Factor Congruence Statistics for C = 85085

Column	Definition
Loop Point	Smallest (magnitude) odd number in the loop sequence.
gcd	Greatest Common Divisor of the odd numbers in the loop sequence. Composite of the factors of C that DO NOT cancel anything in the denominator of the Crossover Point Function (X-Y). Gcd*divisor = 85085.
divisor	Composite of the factors of C that DO cancel factors in the denominator of the Crossover Point Function (X-Y). Gcd*divisor = 85085.
P	From the Hailstone Function, X is always a power of 2. P is the exponent, i.e., $X = 2^{**}P$. P is the sum of elements in the Sequence Vector, for [1,2,3,4], P = 10
Q	From the Hailstone Function, Y is always a power of 3. Q is the exponent, i.e., $Y = 3^{**}Q$. Q is the count of elements in the Sequence Vector, for [1,2,3,4], Q = 4
ccP	Congruence class of $2^{**}P \pmod{\text{divisor}}$. For Factor Congruence matching, ccP must equal ccQ.
ccQ	Congruence class of $3^{**}Q \pmod{\text{divisor}}$. For Factor Congruence matching, ccP must equal ccQ.
cclenP	Length of the set of congruence class of $2^{**}P \pmod{\text{divisor}}$. Congruence class repeats at this frequency. $ccP = cclenP * n + gen1P$ for $n=0,1,2,3\dots$
cclenQ	Length of the set of congruence class of $3^{**}Q \pmod{\text{divisor}}$. Congruence class repeats at this frequency. $ccQ = cclenQ * n + gen1Q$ for $n=0,1,2,3\dots$
gen1P	The first legal power of 2 that is congruence class ccP. (P=0 is invalid for Sequence Vectors)
gen1Q	The first legal power of 3 that is congruence class ccQ. (Q=0 is invalid for Sequence Vectors)
genP	Congruence class generation: IF $P > cclenP$ THEN $genP = (P - gen1P) / cclenP$ ELSE $genP = 1$
genQ	Congruence class generation: IF $Q > cclenQ$ THEN $genQ = (Q - gen1Q) / cclenQ$ ELSE $genQ = 1$
notes	Where noted, Factor Congruence matching does not apply. These are the loops (+C -C -5C -17C) that are independent of C. No factors are cancelled in the Crossover Point Function and as a result, the gcd for these will be 85085.

Loop Point	gcd	divisor	P	Q	ccP	ccQ	cclenP	cclenQ	gen1P	gen1Q	genP	genQ	notes
-85085	85085	1	1	1	2	3							-C
85085	85085	1	2	1	0	3							+C
17017	17017	5	3	1	3	3	4	4	3	1	1	1	
-425425	85085	1	3	2	0	0							-5C
6545	6545	13	4	1	3	3	12	3	4	1	1	1	
60775	12155	7	4	2	2	2	3	6	1	2	2	1	
-146965	7735	11	4	3	5	5	10	5	4	3	1	1	
391391	17017	5	5	3	2	2	4	4	1	3	2	1	
323323	17017	5	5	3	2	2	4	4	1	3	2	1	
7735	7735	11	6	2	9	9	10	5	6	2	1	1	
10829	1547	55	6	2	9	9	20	20	6	2	1	1	
-365365	5005	17	6	4	13	13	8	16	6	4	1	1	
-325325	5005	17	6	4	13	13	8	16	6	4	1	1	
5005	5005	17	7	2	9	9	8	16	7	2	1	1	
3575	715	119	7	2	9	9	24	48	7	2	1	1	
7865	715	119	7	2	9	9	24	48	7	2	1	1	

41327	2431	35	8	4	11	11	12	12	8	4	1	1
31603	2431	35	8	4	11	11	12	12	8	4	1	1
1485715	6545	13	8	5	9	9	12	3	8	2	1	2
1695155	6545	13	8	5	9	9	12	3	8	2	1	2
1878415	6545	13	8	5	9	9	12	3	8	2	1	2
1642795	6545	13	8	5	9	9	12	3	8	2	1	2
1852235	6545	13	8	5	9	9	12	3	8	2	1	2
1380995	6545	13	8	5	9	9	12	3	8	2	1	2
-1446445	85085	1	11	7	0	0						
1547	1547	55	12	4	26	26	20	20	12	4	1	1
23375	935	91	12	6	1	1	12	6	12	6	1	1
55165	935	91	12	6	1	1	12	6	12	6	1	1
-293293	1001	85	12	8	16	16	8	16	4	8	2	1
100555	7735	11	14	8	5	5	10	5	4	3	2	2
10115	595	143	16	7	42	42	60	15	16	7	1	1
17255	595	143	16	7	42	42	60	15	16	7	1	1
3757	221	385	18	6	344	344	60	60	18	6	1	1
-222365	715	119	18	12	106	106	24	48	18	12	1	1
2387	77	1105	20	8	1036	1036	24	48	20	8	1	1
1925	385	221	20	8	152	152	24	48	20	8	1	1
1771	77	1105	20	8	1036	1036	24	48	20	8	1	1
-1209533	221	385	22	14	114	114	60	60	22	14	1	1
-763997	221	385	22	14	114	114	60	60	22	14	1	1
-1274065	1105	77	22	14	37	37	30	30	22	14	1	1
-1092845	1105	77	22	14	37	37	30	30	22	14	1	1
-1153841	221	385	22	14	114	114	60	60	22	14	1	1
-937261	221	385	22	14	114	114	60	60	22	14	1	1
-1287325	1105	77	22	14	37	37	30	30	22	14	1	1
19261	187	455	24	12	1	1	12	12	12	12	2	1
24871	1309	65	24	12	1	1	12	12	12	12	2	1
857395	6545	13	24	15	1	1	12	3	12	3	2	5
89375	715	119	25	14	2	2	24	48	1	14	2	1
2275	455	187	26	12	174	174	40	80	26	12	1	1
19565	455	187	26	12	174	174	40	80	26	12	1	1
46291	119	715	28	16	146	146	60	60	28	16	1	1
63427	1547	55	28	16	36	36	20	20	8	16	2	1
115115	5005	17	31	18	9	9	8	16	7	2	4	2
19591	143	595	32	16	256	256	24	48	8	16	2	1
4147	143	595	32	16	256	256	24	48	8	16	2	1
697	17	5005	36	12	911	911	60	60	36	12	1	1
833	119	715	36	12	196	196	60	60	36	12	1	1
1751	17	5005	36	12	911	911	60	60	36	12	1	1
4369	17	5005	36	12	911	911	60	60	36	12	1	1
1105	1105	77	38	16	25	25	30	30	8	16	2	1
16445	715	119	39	18	43	43	24	48	15	18	2	1
5083	221	385	40	20	331	331	60	60	40	20	1	1
7	7	12155	44	8	6561	6561	120	240	44	8	1	1
8041	187	455	48	24	1	1	12	12	12	12	4	2
5797	187	455	48	24	1	1	12	12	12	12	4	2
935	935	91	48	24	1	1	12	6	12	6	4	4
6643	91	935	52	24	356	356	40	80	12	24	2	1
51947	7	12155	56	32	11306	11306	120	240	56	32	1	1
37835	35	2431	56	32	1582	1582	120	240	56	32	1	1
45157	7	12155	56	32	11306	11306	120	240	56	32	1	1
24059	7	12155	56	32	11306	11306	120	240	56	32	1	1
36197	7	12155	56	32	11306	11306	120	240	56	32	1	1
28651	7	12155	56	32	11306	11306	120	240	56	32	1	1
12803	7	12155	56	32	11306	11306	120	240	56	32	1	1
44765	35	2431	56	32	1582	1582	120	240	56	32	1	1
11515	35	2431	56	32	1582	1582	120	240	56	32	1	1
73339	7	12155	56	32	11306	11306	120	240	56	32	1	1
38171	7	12155	56	32	11306	11306	120	240	56	32	1	1
6307	119	715	56	32	581	581	60	60	56	32	1	1
12047	7	12155	56	32	11306	11306	120	240	56	32	1	1
51023	7	12155	56	32	11306	11306	120	240	56	32	1	1
42595	35	2431	56	32	1582	1582	120	240	56	32	1	1
34741	7	12155	56	32	11306	11306	120	240	56	32	1	1
15385	85	1001	60	30	1	1	60	30	60	30	1	1
6035	85	1001	60	30	1	1	60	30	60	30	1	1
1859	143	595	64	32	86	86	24	48	16	32	3	1
715	715	119	75	42	8	8	24	48	3	42	4	1
10549	77	1105	76	40	16	16	24	48	4	40	4	1
11765	65	1309	82	44	191	191	120	240	82	44	1	1
3179	187	455	84	48	1	1	12	12	12	12	7	4
23647	221	385	84	48	71	71	60	60	24	48	2	1
1331	11	7735	96	48	1	1	24	48	24	48	4	1
8327	11	7735	96	48	1	1	24	48	24	48	4	1

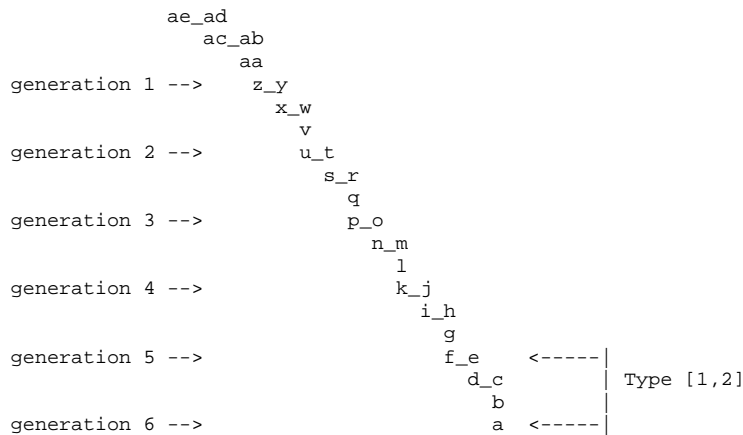
9449	11	7735	96	48	1	1	24	48	24	48	4	1
9515	55	1547	96	48	1	1	24	48	24	48	4	1
5027	11	7735	96	48	1	1	24	48	24	48	4	1
3091	11	7735	96	48	1	1	24	48	24	48	4	1
2365	55	1547	96	48	1	1	24	48	24	48	4	1
605	55	1547	96	48	1	1	24	48	24	48	4	1
889	7	12155	100	40	8856	8856	120	240	100	40	1	1
7007	1001	85	100	56	16	16	8	16	4	8	13	4
8015	35	2431	112	64	1225	1225	120	240	112	64	1	1
7021	119	715	112	64	81	81	60	60	52	4	2	2
9667	7	12155	112	64	3656	3656	120	240	112	64	1	1
19859	7	12155	112	64	3656	3656	120	240	112	64	1	1
1463	77	1105	116	56	1036	1036	24	48	20	8	5	2
781	11	7735	120	48	1	1	24	48	24	48	5	1
9823	11	7735	132	72	4096	4096	24	48	12	24	6	2
13321	77	1105	132	72	781	781	24	48	12	24	6	2
4165	595	143	140	80	100	100	60	15	20	5	3	6
4009	1	85085	156	72	65976	65976	120	240	36	72	2	1
1469	13	6545	156	72	526	526	120	240	36	72	2	1
685	5	17017	156	72	14925	14925	120	240	36	72	2	1
2063	1	85085	156	72	65976	65976	120	240	36	72	2	1
1355	5	17017	156	72	14925	14925	120	240	36	72	2	1
1567	1	85085	156	72	65976	65976	120	240	36	72	2	1
1223	1	85085	156	72	65976	65976	120	240	36	72	2	1
2297	1	85085	156	72	65976	65976	120	240	36	72	2	1
3605	35	2431	156	72	339	339	120	240	36	72	2	1
4313	1	85085	156	72	65976	65976	120	240	36	72	2	1
1357	1	85085	156	72	65976	65976	120	240	36	72	2	1
1853	17	5005	156	72	911	911	60	60	36	12	3	2
3529	1	85085	156	72	65976	65976	120	240	36	72	2	1
1933	1	85085	156	72	65976	65976	120	240	36	72	2	1
409	1	85085	156	72	65976	65976	120	240	36	72	2	1
1927	1	85085	156	72	65976	65976	120	240	36	72	2	1
5785	65	1309	156	72	526	526	120	240	36	72	2	1
1267	7	12155	156	72	5201	5201	120	240	36	72	2	1
1327	1	85085	156	72	65976	65976	120	240	36	72	2	1
529	1	85085	156	72	65976	65976	120	240	36	72	2	1
3475	5	17017	156	72	14925	14925	120	240	36	72	2	1
3961	17	5005	156	72	911	911	60	60	36	12	3	2
5447	13	6545	156	72	526	526	120	240	36	72	2	1
1	1	85085	156	72	65976	65976	120	240	36	72	2	1
3731	91	935	160	80	1	1	40	80	40	80	4	1
3029	13	6545	160	80	1871	1871	120	240	40	80	2	1
10855	65	1309	164	88	1138	1138	120	240	44	88	2	1
12563	17	5005	168	96	2731	2731	60	60	48	36	3	2
1699	1	85085	168	96	7736	7736	120	240	48	96	2	1
21347	1	85085	168	96	7736	7736	120	240	48	96	2	1
27637	1	85085	168	96	7736	7736	120	240	48	96	2	1
539	77	1105	188	104	1036	1036	24	48	20	8	8	3
3839	11	7735	192	96	1	1	24	48	24	48	8	2
1003	17	5005	240	120	1	1	60	60	60	60	4	2
3479	7	12155	268	136	4436	4436	120	240	28	136	3	1
341	11	7735	288	144	1	1	24	48	24	48	12	3
793	13	6545	320	160	5611	5611	120	240	80	160	3	1
2881	1	85085	324	168	50506	50506	120	240	84	168	3	1
3005	5	17017	324	168	16472	16472	120	240	84	168	3	1
457	1	85085	324	168	50506	50506	120	240	84	168	3	1
3305	5	17017	324	168	16472	16472	120	240	84	168	3	1
1721	1	85085	324	168	50506	50506	120	240	84	168	3	1
4447	1	85085	324	168	50506	50506	120	240	84	168	3	1
1193	1	85085	324	168	50506	50506	120	240	84	168	3	1
899	1	85085	492	264	4096	4096	120	240	12	24	5	2

Appendix B

ith, kth Generation Type [1,2] Mersenne Hailstones

- A hailstone is the last number of a partial Collatz sequence (for the sequence 27, 82, 41, 124, 62, 31, the hailstone is 31).
- A Mersenne Hailstone is any hailstone that is Mersenne Number (a power of 2 minus one, such as 31).
- Type [1,2] means the hailstone is preceded by a [1,2] Sequence Vector.
- *kth* Generation means the Sequence Vector is repeated k times.
- *ith* identifies which of the infinite number of kth Generation solutions.
- 31 is the 1st, 1st Generation Type [1,2] Mersenne Hailstone.

In the following diagram, **a** is a 6th generation Type [1,2] Mersenne Hailstone.



The 1st such hailstone, found by plugging (1,6) into the blueprint results in:

$$a = 2^{177149} - 1$$

In binary, **a** has 177,149 contiguous 1 bits. In decimal, it has 53,328 digits. Starting from the hailstone, the Collatz sequence takes 2,386,509 iterations to reach 1. Because **a** is a Mersenne Number, the sequence is predicted to contain

$$m + (1.585 * m)/0.415 \quad \text{or} \quad 4.819 * m$$

iterations of $3n+1$, where m is the number of bits in the hailstone.

Therefore, the sequence should have

$$4.819 * 177,149 \quad \text{or} \quad 853,681$$

iterations of $3n+1$. It actually has

$$854,697$$

making the prediction accuracy 99.88%.

Notes and References:

1) Collatz Problem

From Mathworld

<http://mathworld.wolfram.com/CollatzProblem.html>

2) Collatz conjecture

From Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Collatz_conjecture

3) the Collatz Tree

<http://members.aol.com/mensanator666/treemap/mainmap.htm>

4) Linear Congruence

From Mathworld

<http://mathworld.wolfram.com/LinearCongruenceEquation.html>

5) Linear Congruence

From Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Linear_congruence

6) Euclidean Algorithm

From Mathworld

<http://mathworld.wolfram.com/EuclideanAlgorithm.html>

7) Extended Euclidean Algorithm

From Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Extended_Euclidean_Algorithm

8) Modular Inverse

From Mathworld

<http://mathworld.wolfram.com/ModularInverse.html>

9) Multiplicative Inverse

From Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Multiplicative_inverse

10) Greatest Common Divisor

From Mathworld

<http://mathworld.wolfram.com/GreatestCommonDivisor.html>

11) Greatest Common Divisor

From Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Greatest_common_divisor

12) Hailstone Function Parameters Algorithm

http://members.aol.com/mensanator666/collatz_utilities/python_code.htm#hfp

13) Partition Generator

http://members.aol.com/mensanator666/collatz_utilities/python_code.htm-partitions

14) Collatz_C program

http://members.aol.com/mensanator666/collatz_utilities/python_code.htm#collatz_c