Entropy Makes the Collatz Sequence Go Down

Bradley Berg
Author
bb@techneon.com

Erik Slader
Editor
eslader3@gmail.com

Abstract

We look at the Collatz Sequence from an information theory perspective to lay out its underlying computational mechanics. The mechanisms are similar to those used in pseudo random number generators and one way hashes. An equivalent restatement of the Collatz sequence steps through alternating chains of even and odd values. This variation constitutes a pseudo random number generator. The operations used to scramble values are unbiased which results in an even distibution of ones and zeros. The entropy of this mechanism is high so that in the second phase values are fairly randomized.

1. Introduction

The computational mechanics of the Collatz sequence are analyzed to determine the odds of taking an even or odd step. With the following definition of the sequence we show the the average odds of taking either step are even. In this case statistically the sequence will on average decline and eventually terminate. This variation is often referred to as the Syracuse sequence.

$$\text{N is Even:} \quad N' = \frac{N}{2}$$
$$\text{N is Odd:} \quad N' = \frac{3N+1}{2}$$

The gain of a each transition is its Output divided by the Input (N' / N). As N gets larger in the odd transition the "+ 1" term quickly becomes insignificant. To compute the gain for the odd transition in the limit we can safely drop the "+1"term.

	Output / Input	Gain
N is Even:	$\frac{N}{2N}$	0.5
N is Odd:	$\frac{3N+1}{2N}$	1.5

The total gain of a series of transitions is the product of the gains for each transition. Based on this the average gain of a sequence depends on the probability of taking either an odd or even path.

$$G_s = 1.5^{p_{\text{odd}}} \cdot 0.5^{p_{\text{even}}}$$

The choice of which path is taken is determined by the low order bit of the input value. If the sequence should produces uniformly randomized values then the chances of taking either transitions is 50:50. This implies the low bit would need to be uniformally random over the sequence. The average gain of a uniformly randomized sequence is then:

$$\text{Average transaction gain} = \langle \text{transaction gain} \rangle^{p(\langle \text{transaction} \rangle)}$$
$$\text{Average odd gain} = 1.5^{0.5} = 1.22474$$
$$\text{Average even gain} = 0.5^{0.5} = 0.70711$$
$$\text{Average sequence gain: } G_a = 1.22474 \cdot 0.70711 = 0.86603+$$

The statistical average gain in each step is less than one so on average the sequence declines. However, the gain in a single run for a given Seed can vary significantly. There could potentially still be a series where the the total gain indefinitely exceeds one and never terminates.

$$\begin{aligned}
\text{Breakeven Gain} &= 1.5^p \cdot 0.5^{1-p} = 1 \\
\ln\left(1.5^p \cdot 0.5^{1-p}\right) &= \ln(1) = 0 \\
&= \ln(1.5^p) + \ln(0.5^{1-p}) = 0 \\
&= p \cdot \ln(1.5) + (1-p) \cdot \ln(0.5) = 0 \\
p \cdot 0.40546 &= -(1-p) \cdot (-0.69315) \\
\frac{0.40546}{-0.69315} &= \frac{-(1-p)}{p} \\
-0.58497 &= -\frac{1-p}{p} \\
-0.58497 - 1 &= -\frac{1}{p} \\
p &= \frac{1}{1.58497} \approx 0.63093
\end{aligned}$$

$$\text{Gain} = 1.5^{0.63} \cdot 0.5^{0.37} \approx 0.99898 \quad \text{Under breaking even}$$

$$\text{Gain} = 1.5^{0.64} \cdot 0.5^{0.36} \approx 1.01001 \quad \text{Over breaking even}$$

For any series of values to continually increase and never terminate it would have to sustain an average gain over one. To break even odd transitions would need to occur about 64% of the time. They would need to be applied over 1.7 times more than evens; which is substantially skewed. It remains to be shown that the sequence does not intrinsically favor odd transitions.

[Copyright notice will appear here once 'preprint' option is removed.]

1.1 Even and Odd Chains

Consecutive iterations of the same kind of transition in a run form a chain. Even chains start with an even seed value that in binary have one or more trailing zeroes. After applying the transitions in an even chain the result simply has the low order zeros removed.

Odd chains consume an odd input and have multiple odd intermediate values. Eventually an Odd chain transitions to an even number. The number of consecutive low order one bits determines the chain length. For example, an input of 19 is a binary 10011 so the subsequent chain has two Odd transitions: $19 \rightarrow 29 \rightarrow 44$

Let k be the number of low order one bits and j is the input value with the low one bits removed plus 1.

The input to an odd chain has the form: $j \cdot 2^k - 1$

The output of the chain simplifies to: $j \cdot 3^k - 1$

$$
\begin{aligned}
N_1 &= \frac{3N+1}{2} && \text{First transition} \\
&= \frac{3(j \cdot 2^k - 1) + 1}{2} && \text{Substitute } N = j \cdot 2^k - 1 \\
&= \frac{3j \cdot 2^k - 3 + 1}{2} \\
&= \frac{3j \cdot 2^k - 2}{2} \\
&= 3^1 \cdot j \cdot 2^{k-1} - 1
\end{aligned}
$$

$$
\begin{aligned}
N_{i+1} &= \frac{3\left(3^i \cdot j \cdot 2^{k-i} - 1\right) + 1}{2} && \text{Subsequent transitions} \\
&= \frac{9 \cdot j \cdot 2^{k-i} - 3 + 1}{2} \\
&= \frac{9 \cdot j \cdot 2^{k-i} - 2}{2} \\
&= 3^{i+1} \cdot j \cdot 2^{k-i-1} - 1 \\
N_k &= 3^k \cdot j - 1 && \text{Odd chain output}
\end{aligned}
$$

Each run of the Collatz sequence will have segments with alternating even and odd chains. For reference, Table 1 lists the first few chains for the series beginning with a seed of 27.

Syracuse Sequence	Even	Odd	j	k
$27 \rightarrow 41 \rightarrow 124 \rightarrow 62$		$27 \rightarrow 62$	7	2
31	$\rightarrow 31$		31	1
$47 \rightarrow 71 \rightarrow 107 \rightarrow 161 \rightarrow 242$		$\rightarrow 242$	1	5
121	$\rightarrow 121$		121	1
$364 \rightarrow 182$		$\rightarrow 182$	61	1
91	$\rightarrow 91$		91	1
$137 \rightarrow 206$		$\rightarrow 206$	23	2
103	$\rightarrow 103$		103	1
$155 \rightarrow 233 \rightarrow 350$		$\rightarrow 350$	13	3
175	$\rightarrow 175$		175	1
$263 \rightarrow 395 \rightarrow 593 \rightarrow 890$		$\rightarrow 890$	11	4

Table 1. Partial Syracuse Run Seed = 27

1.2 Combining Even And Odd Chains

Each run alternates between even and odd chains. We represent this algebraically by merging both into a single step. This gives us a series defined as a single transition. Since all intermediate values using this combined definition are even, an initial odd value needs to first transition one step using the "3n + 1" rule to reach the first even number.

Every input has the binary form: [j] [ones] [zeros]

$$N = ([j+1] \cdot 2^{k_o} - 1) \cdot 2^{k_z} \rightarrow [j+1] \cdot 3^{k_o} - 1$$

where:

- k_z - number of trailing zeros in N
- k_o - number of the next higher set of ones in N
- j - N shifted right by $k_z + k_o$ bits: $\frac{N}{2^{k_z+k_o}}$

Using the previous example, initially we transition 27 to 82. From there the next few steps are in Table 2.

	j	ko	kz	binary
$82 \rightarrow 41 \rightarrow 124 \rightarrow 62$	20	1	2	10100_10
$31 \rightarrow 484 \rightarrow 242$	0	5	1	0_111110
$121 \rightarrow 364 \rightarrow 182$	60	1	1	111100_10
$91 \rightarrow 274 \rightarrow 206$	22	2	1	10110_110
$103 \rightarrow 310 \rightarrow 466$	12	3	1	1100_1110
$233 \rightarrow 700 \rightarrow 350$	116	1	1	1110100_10
$593 \rightarrow 1780 \rightarrow 890$	10	4	1	1010_11110

Table 2. Combined Even And Odd Chains

2. Sequence Entropy

Statistical averages only hold when the odds are fair. In this section, we show why the dice are not loaded. Shannon entropy is a measure of information denoting the level of uncertainty about the possible outcomes of a random variable [1].

$$H = -p0 \cdot \log_2(p0) - p1 \cdot \log_2(p1)$$

where:

- $p0$ is the probability a bit is zero.
- $p1$ is the probability a bit is one $(1 - p0)$.

A set of coin tosses has $p0 = p1 = 0.5$ so its entropy is 1; totally random. When looking at the entropy of bits in a number then $p0$ is the percentage of zero bits. For the binary number, 1010_1111, $p0$ is 0.25 ($H = 0.811$). Strings of all ones or zeros have no entropy ($H = 0$). For a binary number, we are measuring the bits in a number horizontally.

Bits in a series of numbers have two dimensions - horizontal bits in each individual value and vertical bits over the duration of the series. We can also measure entropy vertically over a number series. That means we can observe a select bit position in each value as the series progresses.

For Collatz, the low-order bit is of interest because it determines if a number is odd or even. In turn, that determines which transition to take. When the entropy of the low-order bit is high, then on average there are nearly as many even transitions as odds.

Each kind of chain takes a value where the low-order bits are a string of zeros or ones and either removes or replaces them. Even inputs remove low-order zeros. The expression for odd chain inputs has a 2^k term that transitions to a 3^k term.

Since strings of zeros have no entropy and the j term has positive entropy, entropy increases each time an even transition is applied. In odd transitions, entropy is also increased by removing the repeated ones and again by scrambling the remaining bits. The upper bits, j, are scrambled by multiplying j by a power of 3. As a run progresses, this increase in entropy randomizes the values. The number of odd and even transitions balance out driving the sequence downward and eventually forcing it to terminate.

2.1 Losing Information

A Seed can be contrived to produce a run of any desired length. The longer the run, the larger the seed has to be in order to contain enough information to influence the desired outcome. Initially, as a run progresses, the information contained in the Seed is lost. When there are two possible ways to reach a value in a run we lose the information about which path was taken to reach it [2].

Odd numbers always transition 3n + 1 to even numbers, so an odd value can only be reached from one even value. However, certain even numbers can be reached from either an odd or even transition. For example, an output of 16 can be reached from either 32 or 5:

$$32 \rightarrow 32/2 = 16$$
$$5 \rightarrow 3 \cdot 5 + 1 = 16$$

Whenever transitioning to an even value such that $(\text{even}\%3 = 1)$, then the previous value could have been either:

$$2 \cdot \text{even or } (\text{even} - 1)/3$$

For Collatz, a bit of information contained in the seed is lost each time one of these select even numbers is reached. After all bits in the seed are scrubbed, this initial phase is complete. Any attempt to contrive a Seed to skew results can only directly affect values during this phase.

One-way hash functions rely on this concept of lost information [3]. Secure hashes have many ways to reach each hashed value. This is how passwords are encoded and used for authentication. Using this metaphor, you can think of the Seed as a password and the Collatz sequence as a trivial one-way hash schema used to mask it.

2.2 RandomizationPhase

This next phase is key as this is where the sequence runs below the Seed. The sequence is rewritten as a pseudo random number (PRNG) generator. Hastad et al. (1999) [4] show that any one-way hash can be used to create a PRNG. Uniformly randomized values eventually trend towards their average. In turn, this drives transitions towards their average gain. In the introduction, we showed that Collatz has an average gain of 0.86603, eventually driving the series below the Seed value.

We'll be using the combined series from section 1.2 for each randomization step. The value, j, is always even so the $[j+1]$ term will simply set the low order bit to one as there is no carry. Also, the product will have an odd result so that decrementing by 1 will likewise just clear the low order bit.

$$\text{Input: } ([j+1] \cdot 2^{k_o} - 1) \cdot 2^{k_z}$$

$$\text{Result: } [j+1] \cdot 3^{k_o} - 1 = [j \oplus 1] \cdot 3^{k_o} \oplus 1$$

In Table 3, the top line has steps for a randomization phase that begins with 647. Calculations for each combined transition $(1942, 2186, 1640, 308, 116)$ are shown in binary:

A pseudo-random number generator shown in Figure 1 repeatedly applies a function to produce a series of values. In order to produce uniformly random numbers, operators cannot be biased towards producing either more ones or zeros. In a uniform sequence, the entropy will be one. If it is not uniform, the bias will show up in the operators. w **Select**

If you remove some low-order bits of a random number, the remaining part will still be random. Using the upper bits from the input still gives random values. However, the way the value is split, the selected upper value will be even. The low-order bit is zero, and only the other bits are a randomized portion.

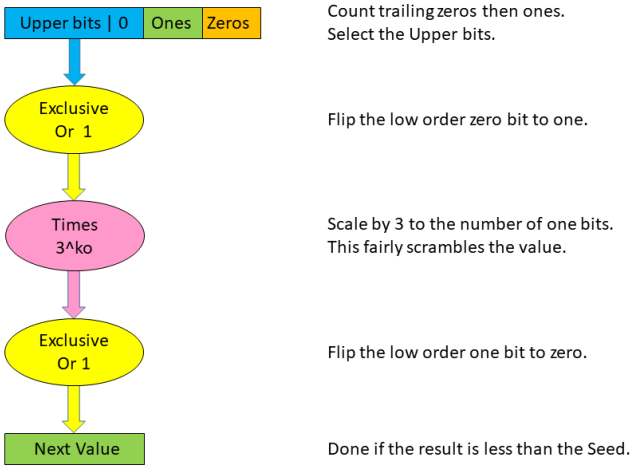

Figure 1. Sequence Redefined As A Hash Function

	1942	**2186**
Input	11110010_110	1000100010_10
Shift Right	11110010	1000100010
Xor 1	11110011	1000100011
Times 3^{ko}	100010001011	11001101001
Xor 1	1000100010_10	1100110_1000
	1640	**116**
Input	1100110_1000	100110_100
Shift Right	1100110	100110
Xor 1	1100111	100111
Times 3^{ko}	100110101	1110101
Xor 1	100110_100	1110100

Table 3. Hash Function Example In Binary

Logical Exclusive Or 1

The first Exclusive Or sets the low bit of the selected region. This is balanced out by clearing it in the final step with another Exclusive.

Product

The product of a random variable by a constant is also random, but with a larger gap between them. Multiplying random numbers from 1 to 10 by 3 yields random numbers from 3 to 30. They simply have a gap of 3 between them instead of 1.

The product used to scramble values is equivalent to repeated sums of the input. Table 4 shows all combinations for the three inputs (A, B, Carry In) and the two outputs (Sum, Carry Out). It also shows changes (Exclusive Or) between the sum and inputs A and B:

Input bits A and B are vertically aligned and are altered by addition. Carries are applied horizontally and propagate to higher order bits. This way, bits in both directions become scrambled.

Note that all the columns in the table are different. This shows how bits are scrambled to produce randomized results. Also note that all columns contain 4 zeros and 4 ones. This balance produces results that are unbiased towards either zero or one bits. The end result is a series of uniformly distributed pseudo-random numbers.

Outputs in any individual series depend on the values k_z and k_o. The more random they are, the more random the series. The k values measure the width of a horizontal subset of bits in each

A	B	Carry In	Sum	Carry Out	A ⊕ Sum	B ⊕ Sum
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	0	1
1	0	1	0	1	1	0
1	1	0	0	1	1	1
1	1	1	1	1	0	0

Table 4. Bitwise Addition

value. Pseudo-random number generators that conflate operations on horizontal and vertical sets of bits rely on the independence of these orthogonal values.

The repeated zeros and upwardly ones in the lowest bits that might have low entropy are continuously removed and replaced with scrambled bits. This creates a self-regulating system that continuously randomizes the lower bits. Since those bits control the selection operation in the next round.

When runs have uniformly random values, then revisiting the Syracuse sequence, the average number of even and odd transitions will balance out. In turn, this causes the run to decrease since the average gain is less than one. If the sequence was not uniformly random, then we would see bias amongst the arithmetic operations used in each round.

Examples where seeds produce long runs will have highly skewed values in the first phase, but that cannot be sustained. As values become more randomized and the series progresses, they will trend towards average results. With coin tossing, even if you get lucky and call the results of several coin tosses, your luck will run out in the long run.

2.3 Reduction To One

The previous randomization phase leaves us with a value of N that is below the seed. It is well understood that once this happens, we know the series will eventually terminate at one. Firstly, we know all values below some arbitrary small number M (say 10) transition to one.

Next, starting with the next higher Seed, $M + 1$, we transition until it reaches M or less. Since we already know Seeds of M or less will reach one, by induction, once a series goes below its Seed we know it will reach one. This is why even Seeds are uninteresting as they immediately decline.

When measuring the length of a run, including this phase is not useful and can distort any result. When winding down as numbers get smaller, they can become more irregular. Instead of defining the run length as the number of steps to one, use only the number of odd steps until the series goes below the Seed. It is usually more practical to count only odd steps because it corresponds to the number of terms in the algebraic expansion of a run.

2.4 Observed Entropy

The first few values will have lower entropy until enough bits are included to average out. In the Introduction, we've shown that to sustain an infinite run there needs to be 64% or more ones. This gives an entropy of:

$$H = -0.36 \cdot \log_2(0.36) - 0.64 \cdot \log_2(0.64) = 0.94268$$

Individual runs will typically have some jitter since we are performing discrete computations. There will be higher entropy at the end of very long runs, which are rare. In the first phase, long runs will be skewed towards more odd steps to make the values grow larger up front. Short runs where evens dominate won't even make it to the randomization phase.

To compute the entropy, the low-order zero bit is discarded as it is fixed. Also, since the values have a variable width, the uppermost bit where one is also discarded. This differs from practical PRNGs where the values have a fixed width.

To see the randomization in action, Table 5 lists the entropy for the first two phases. Entropy is computed using the accumulated number of ones and zeros in the run. The counts of ones and zeros are reset at the start of the Randomization phase so that those computations are completely separate. Even in the Information Loss phase, entropy is well above the 0.94268 bound right out of the gate. The computed length of the Information Loss phase is quite conservative.

Seed $= 4\_50449\_75045\_09599 = 10\_00d1\_0da5\_de9f_{base16}$

Step	Entropy	Ones	Zeros	Notes
1	0.99750	24	27	Information
2	0.99993	52	51	Loss
3	0.99988	77	79	phase 1
4	0.99998	105	104	
5	0.99934	136	128	
6	0.99965	163	156	
7	0.99950	194	184	
8	0.99986	222	216	
9	0.99999	250	248	
10	0.99994	281	276	
11	0.99973	314	302	
12	0.99984	342	332	
13	0.99993	369	362	
14	0.99966	402	385	
15	0.99983	428	415	
16	0.99989	456	445	
17	0.99982	487	472	
18	0.99975	518	499	
19	0.99988	545	531	
20	0.99920	31	29	Randomization
30	1.00000	321	321	phase 2
40	0.99978	606	585	
50	0.99966	899	861	
60	0.99983	1192	1156	
70	0.99969	1489	1429	
80	0.99996	1770	1744	
90	0.99963	2105	2012	
100	0.99967	2410	2309	
110	0.99876	2772	2551	
120	0.99828	3105	2816	
130	0.99890	3387	3133	

Table 5. Example Of Entropy In A Run

If the hash function had a bias, it would show up by running it over many consecutive numbers. Here the hash was run over a million consecutive even numbers. Table 6 shows the cumulative entropy of the resulting values, which is very near one as expected.

Iteration	Entropy	Ones	Zeros
50,000	0.99350	475,206	574,794
100,000	0.99721	984,678	1,115,322
150,000	0.99886	1,512,405	1,637,595
200,000	0.99966	2,054,339	2,145,661
250,000	0.99984	2,585,879	2,664,121
300,000	0.99970	3,086,254	3,213,746
350,000	0.99987	3,625,941	3,724,059
400,000	0.99970	4,113,665	4,286,335
450,000	0.99976	4,638,681	4,811,319
500,000	0.99983	5,168,585	5,331,415
550,000	0.99991	5,711,156	5,838,844
600,000	0.99996	6,255,089	6,344,911
650,000	1.00000	6,816,718	6,833,282
700,000	0.99994	7,415,041	7,284,959
750,000	1.00000	7,882,886	7,867,114
800,000	1.00000	8,392,173	8,407,827
850,000	1.00000	8,925,576	8,924,424
900,000	1.00000	9,428,185	9,471,815
950,000	0.99999	9,944,542	10,005,458
1,000,000	0.99999	10,466,775	10,533,225
1,048,576	1.00000	11,012,891	11,007,205

Table 6. Entropy Of The Hash Function

Table 7 illustrates a contorted sequence where some bits are artificially forced to one. This shows how skewed the operators have to get in order for entropy to drop below 0.94268, where it would increase indefinitely. 11 out of 20 bits are needed to be forced to one to sufficiently skew the result.

Value Of Forced Bits	Final Entropy	Ones	Zeros
0	1.00000	11,012,891	11,007,205
10,000	0.99995	11,102,306	10,917,790
11,000	0.99971	11,229,280	10,790,816
11,100	0.99931	11,350,975	10,669,121
11,500	0.99916	11,385,687	10,634,409
15,500	0.99892	11,435,329	10,584,767
55,500	0.99703	11,716,482	10,303,614
155,500	0.99727	11,687,638	10,332,458
175,500	0.98541	12,573,465	9,446,631
177,500	0.96792	13,323,124	8,696,972
177,700	0.95400	13,775,529	8,244,567
177,740	0.94266	14,093,550	7,926,546

Table 7. Entropy Of A Contorted Hash Function

3. Conclusion

The Collatz sequence incorporates principle mechanisms commonly used to create pseudo random number generators.

- To overcome a contrived Seed, one way hashing smothes out any regularities.
- Repeated low order one and zero bits are erased at each step.
- A product using independent values randomizes values.

Any individual run is partitioned into three phases. In the initial phase the Seed value can influence the outcome to produce arbitrarily long runs. After that the series generates randomized values until it goes below the Seed value. From there it is guaranteed to reduce to one unless the series is circular. A uniformly randomized series eventually moves towards a statistically average gain. For a Collatz series to sustain an average gain above one would require over 1.7 times more odd transitions than even. This is well above parity. Instead, randomization forces the series to average out and decrease until it inevitably goes below the seed. Once it does that we know it will terminate. The random behavior of the Collatz sequence makes it impossible to prove algebraically. Conway[5] showed that a generalization of the 3N + 1 problem is undecidable. Trying to make sense of the values in the series is akin to analysing values produced by a random number generator. The irony is that this randomness is the force that leads to convergence.

References

[1] Behrouz Zolfaghari, Khodakhast Bibak, and Takeshi Koshiba. The Odyssey of Entropy: Cryptography In *Entropy* 2022, 24(2), 266. https://www.mdpi.com/1099-4300/24/2/266/pdf

[2] John C. Baez, Tobias Fritz, Tom Leinster. A Characterization of Entropy in Terms of Information Loss In *Entropy* 2011, 13(11), 1945-1957 https://mdpi-res.com/d_attachment/entropy/entropy-13-01945/article_deploy/entropy-13-01945-v2.pdf

[3] Russell Impagliazzo, Leonid A. Levin, Michael Luby. Pseudo-random Generation from one-way functions" In *David S. Johnson (ed.),Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, May 14-17, 1989, Seattle, Washington, USA, ACM pp. 12-24, doi:10.1145/73007.73009, S2CID 18587852 https://dl.acm.org/doi/pdf/10.1145/73007.73009

[4] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, Michael Luby. A pseudorandom generator from any one-way function. In *Siam Journal on Computation 28(4):1364-1396*, 1999. https://epubs.siam.org/doi/10.1137/S0097539793244708

[5] John H. Conway. Unpredictable iterations. In *Proc. 1972 Number Theory Conf., Univ. Colorado, Boulder. pp. 49-52.*

A. Appendix

Here are the entropy values for the randomization phase of some long runs. The Sample Size is the number of values produced by the algorithm for randomization. All of the entropy values are well above 0.94268; the entropy required to produce an infinitely long run.

Entropy is computed from values in each run. Since at the end of each step the values are all even, the low order bit is discarded. Values also have variable widths; unlike values in a practical PRNG. To account for this the upper one bit is also discarded. The entropy is then computed using the total number of ones and zeros in each run. To illustrate this the chart shows the entropy after five steps into the randomization phase.

Run Length	Seed	Entropy	Sample Size	Run Run	Seed	Entropy	Sample Size
200	3718_71359	0.99979	68	340	126_76301_41951	0.99994	116
205	1958_82855	0.99980	70	345	1_22350_60455	0.99973	119
210	1274_56255	0.99983	95	350	401_88187_72839	0.99979	129
215	1937_66367	0.99899	67	355	724_40525_17375	0.99801	118
220	19681_65887	0.99995	82	360	825_34095_41119	0.99938	122
225	6560_55295	0.99993	86	365	712_17685_99551	0.99972	132
230	13046_21055	0.99981	83	370	1220_47117_99967	0.99612	120
235	1_35512_07911	0.99998	82	375	2268_60535_77471	0.99917	133
240	32463_39311	0.99929	80	380	8939_48210_46783	0.99935	141
245	90870_90719	0.99952	73	385	17070_06081_85759	0.99994	122
250	18014_87687	0.99891	83	390	14939_44647_67771	0.99451	131
255	1_48843_35615	0.99982	92	395	2636_51570_99263	0.99985	131
260	2_39626_04007	1.00000	92	400	21799_62942_60379	0.99936	136
265	2_52445_54015	1.00000	94	405	12918_29891_91335	0.99939	140
270	27410_96351	0.99950	86	410	5637_56180_56351	0.99930	153
275	5_98341_74399	0.99830	92	415	5939_15976_23151	0.99930	154
280	31365_10111	0.99944	96	420	48746_49581_33967	0.99923	153
285	3_73541_80511	0.99850	96	425	41352_12952_02335	0.99810	151
290	5_01732_47487	0.99918	97	430	1668_08468_87871	0.99928	151
295	21_53848_33215	0.99991	104	435	28943_78244_38015	0.99719	152
300	106_36415_82407	0.99992	102	440	12087_56891_76347	0.99824	158
305	28_43969_52295	0.99929	105	445	6367_11449_15935	0.99826	160
310	2_01366_15963	0.99955	106	450	50656_70902_68923	0.99994	170
315	2_12138_83483	0.99963	108	455	7066_59241_17439	0.99819	163
320	47_83372_65823	0.99977	114	460	8_11790_31161_74975	0.98410	158
325	25_19636_62655	0.99971	117	465	7_84895_27993_66463	0.99336	151
330	225_38353_49759	0.99974	105	470	4_64802_52950_28271	0.99795	151
335	2_61309_34783	0.99950	114	475	5_68559_52882_94911	0.99948	174