Predictable Behaviors of Collatz Chains

Introduction.

The Collatz conjecture is a famous unsolved problem in mathematics. It asks whether, given any positive integer n, n always eventually reach 1 in a finite number of steps via the Collatz map. For each step in this process, we take an input n, mapping it to $n/2$ if n is even, and $3n+1$ if n is odd ---and repeat with the result.

Examples of Collatz "chains":

$12 \to 6 \to 3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1.$

$13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1.$

$21 \to 64 \to 32 \to 16 \to 8 \to 4 \to 2 \to 1.$

The conjecture was initially introduced by Lothar Collatz in 1937. Despite there being hundreds of published papers on the conjecture, and many unpublished failed attempts at proofs, the conjecture remains unsolved. However, we do have some partial results: in particular, a 2017 distributed computing project verified the Collatz conjecture for all starting values of n up to 10^{20}. Furthermore, it was shown in 1993 that any nontrivial "cycle" of numbers obtained via Collatz (i.e. a group of numbers that form a loop and thus never descent to 1, not including the trivial loop $1 \to 4 \to 2 \to 1$) must have length at least 17,087,915. Thus, simply finding a counterexample is extremely unlikely.

In this project, we investigate various patterns that emerge in the iterations of numbers via this Collatz process, as well as provide some generalizations and observations with regard to the behavior of chains under the Collatz process. In particular, we are interested in examining (1) the rate at which a number n decreases in the long term, given its size, and (2) the number of steps any given number takes to reach 1 (and factors that influence this).

Arguments.

Despite the remaining uncertainty regarding when (if ever) this conjecture will be proven (or disproven), we can construct a somewhat convincing heuristic argument for why the Collatz conjecture is *probably* true. Observe:

- 2^n takes exactly $n-1$ steps to reach 1.
- $3n+1$ is always even for n odd, whereas $n/2$ could be even or odd for n even. Thus, we often observe long sequences of division by two, but we can never have consecutive $3n+1$-steps.

As a result, the maximum rate at which a chain can grow is approximately $1.5n$ (every 2 steps).
- For notational convenience, let X represent the event that we take $n/2$, and Y represent the event of taking $3n+1$. Given some large even number N s.t. $2^k \leq N < 2^{k+1}$, notice that $P(X^k) = 1/2^{k-1}$ and $P(X^{k-1}) = 1/2^{k-1}$, as 2^k and $3 \cdot 2^{k-1}$ are two of the 2^{k-1} even numbers in this interval. $P(X^{k-2}) = 2/2^{k-1} = 1/2^{k-2}$, $P(X^{k-3}) = 4/2^{k-1} = 1/2^{k-3}$, $P(X^{k-4}) = 8/2^{k-1} = 1/2^{k-4}$, and so on, until $P(X^1) = 1/2$ (since half of the even numbers are simply two times an odd numbers). Thus the expected number of divisions by two of N is $\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{5}{32} + \cdots + \frac{k-1}{2^{k-1}} + \frac{k}{2^{k-1}}$. This is approximately equal to 2 for k very large, so we can observe that for large values of n, we expect n to be reduced to $0.75n$ three steps later (mulitply by 3, divide by 2 twice). And since $0.75 < 1$, numbers should on average decrease, and thus eventually arive at 1.

However, this is only a probabilistic observation, and is in fact a fairly well-known argument (see Tao's presentation). If one can show that *all* initial values n (aside from one) eventually iterate to a number *less than* n, this would could the Collatz conjecture via induction.

Note that there do exist incomplete proofs of the conjecture; in particular, the ones claimed by Yamada (1981), Cadogan (2006), and Bruckman (2008). Surveys on results on the problem have been published by Jeffery Lagarias (in addition to other authors).

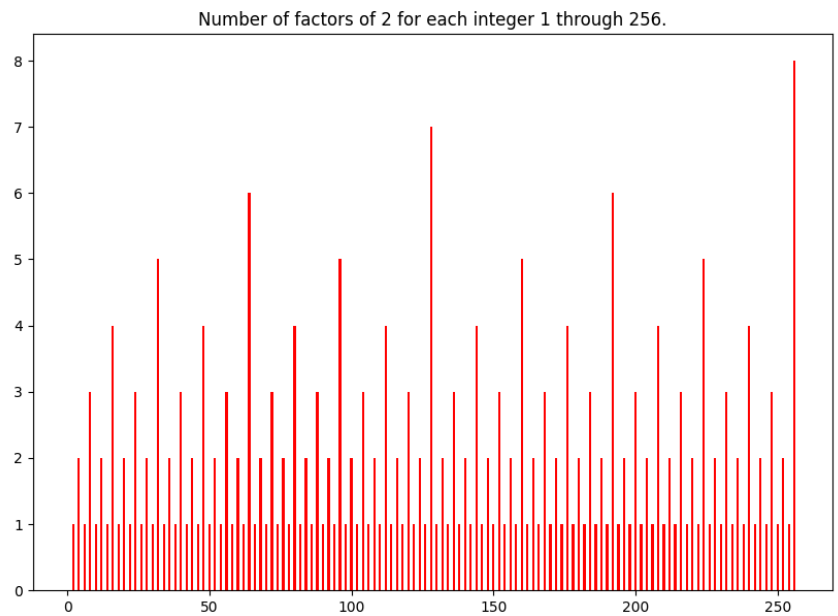
In 2019, Terrence Tao made significant progress on the conjecture in proving that “almost all” — that is, greater than 99% — of numbers eventually fall to a value less than 200 (and thus eventually reach 1). He limited his sample of initial values primarily to numbers that were 1 (mod 3), as numbers that are 2 (mod 3) quickly fall off. To see why, take any odd number n: n mod 6 is 1, 3, or 5, so $(3n+1)$ mod 6 is 4. Divide $(3n+1)$ by 2 — it’s either 2 or 5 mod 6. Next: if the result is even, it’s 1 mod 6; otherwise, it’s 4 mod 6. In other words, it’s 1 mod 3.

An interesting notion when thinking about the Collatz conjecture is the question of how we can thinking about the problem in reverse: that is, given some number n, what numbers fall to / lead to n? This quetion is answerable by the inverse Collatz relation, which outputs $2n$, in addition to $(n-1)/3$ if n mod 6 is 4 (i.e., so that $(n-1)/3$ is an odd whole number). This allows us to construct what is known as a Collatz tree (see Overleaf file).

One observation we find in examining these trees is that only every other power of two (4, 16, 64, etc.) is divisible by 3 mod 1— that is, odd powers of two minus one are not divisible by 3. Things get unpredictable when we branch further out from the root of the tree, however, with some numbers quickly getting large, and other bouncing up and down.

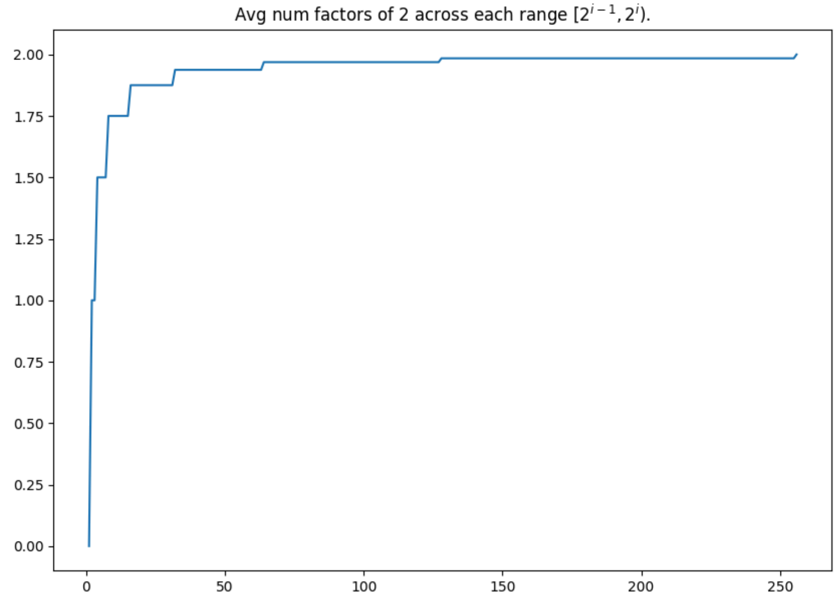
To begin, we illustrate our heuristic argument for the validity of the Collatz conjecture by plot the number of factors of 2 an integer n has in the graph below.

```
import math
import sympy
import matplotlib.pyplot as plt

def numTwos(n):
  sum = 0
  while n%2 == 0:
    sum += 1
    n = n/2
  return sum

X = list(range(1,257))
Y = [numTwos(x) for x in X]

plt.figure(figsize=(10,7))
plt.bar(X, Y, color='red')
plt.title("Number of factors of 2 for each integer 1 through 256.")
plt.show()
```



Now, we plot the average number of prime factors of 2 for each integer n across the range $[2^{i-1}, 2^i)$ from $i = 1$ through 9. As we can see, this value quickly approaches 2 as n gets large, confirming our argument for the expected number of "$n/2$" steps in any Collatz chain to be 2.

```
X = list(range(1,257))
Y = [numTwos(x) for x in X]

for i in range(1,9):
  sum_num_power = 0
  for j in range(2**(i-1),2**i):
    sum_num_power += Y[X[j-1]-1]
  sum_num_power = sum_num_power/2**(i-2)
  for j in range(2**(i-1),2**i):
    Y[X[j-1]-1] = sum_num_power
Y[255] = 2

plt.figure(figsize=(10,7))
plt.plot(X, Y)
plt.title("Avg num factors of 2 across each range $[2^{i-1},2^i)$.")
plt.show()
```

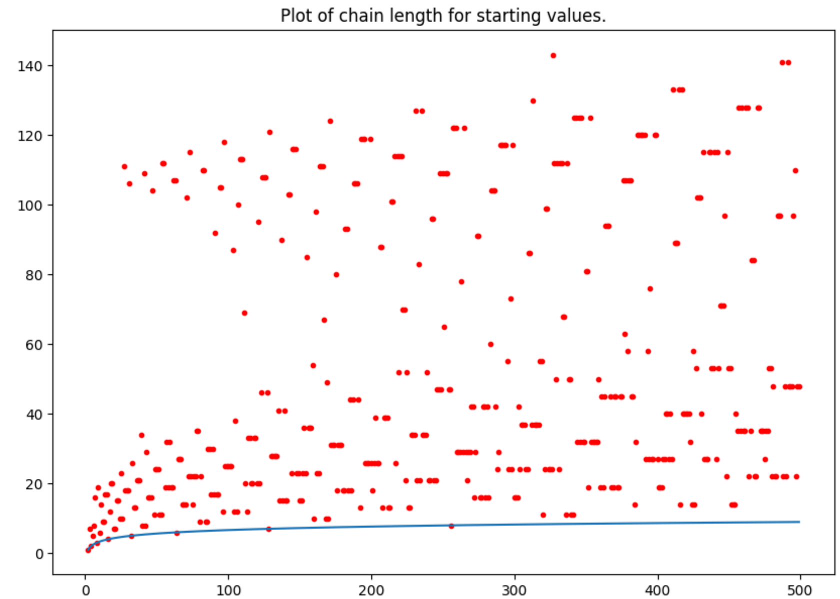


Below, we'll plot some data for Collatz chains and look for patterns. We begin by plotting the number of steps it takes some given starting value n to reach 1. Notice that since powers of two cascade straight down to 1, the the smallest possible number of steps for this is $\log_2(n)$.

```
def collatz(n): # Collatz function (returns n after 1 step)
  if n%2 == 0:
    return n//2
  elif n%2 == 1:
    return 3*n+1

X = []
Y = []
for i in range(2,500): # look at the list of odd numbers from 2 through 500.
  X.append(i)
  steps = 0
```



```
    while(i != 1):
        i = collatz(i)
        steps += 1
    Y.append(steps)

plt.figure(figsize=(10,7))
plt.scatter(X, Y, marker='.', color='red')
plt.plot(X, [sympy.log(x, 2) for x in X])
plt.title("Plot of chain length for starting values.")
plt.show()
```



Now, look at a graph of Collatz chains to visualize how numbers bounce up and down. Additionally, we plot the following curves over top of these chains:

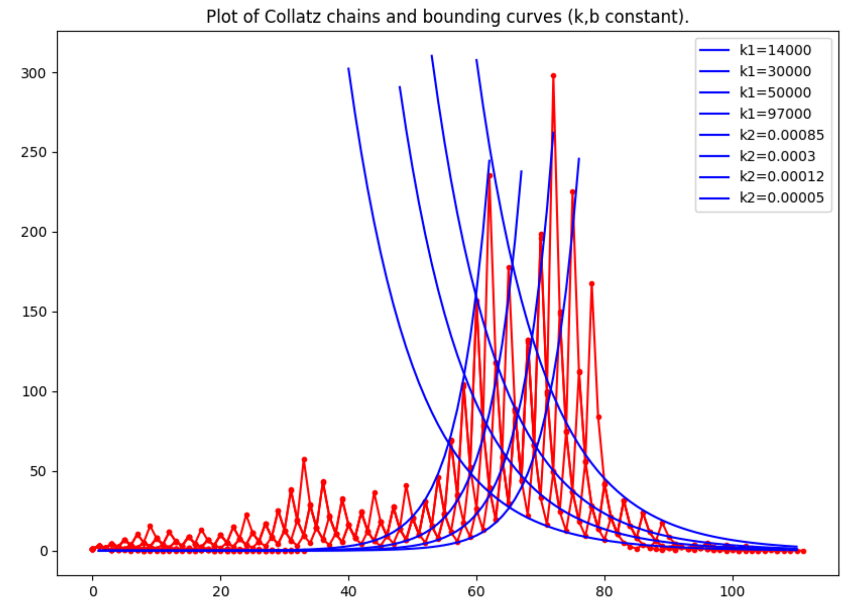
- $y = k_1(3/4)^{x/3}$, where k_1 is some constant. This illustrates the fact that the decrease every 3 steps often tends to be 3/4. The reason we use $x/3$ as the exponent in the functions below (and not x) is because there are 3 steps to go from $n \to n/2 \to n/4 \to 3(n/4) + 1$.

- $y = k_2(3/2)^{x/2}$, where k_2 is some constant. This illustrates that the maximum rate at which a chain can increase is by 3/2 every 2 steps (recalling the fact that $3n + 1$ must necessarily be even, so we can never have consecutive $3n + 1$ steps.

```
startingN = [] # list of duples with [input value, number of steps].
for i in range(30,60): # look at the list of odd numbers between 30 and 60.
  if i%2 == 1:
    initial = i
    steps = 0
    while i != 1:
      i = collatz(i)
      steps += 1
    startingN.append([initial, steps])

plottingN = []
for k in startingN:
  initial = k[0]
  n = initial
  chain = [] # Collatz chains, normalized so each number is divided by initial.
  for i in range(0,k[1]):
    chain.append(float(n/initial))
    n = collatz(n)
  plottingN.append([chain]) # list of chains for each initial value

# Plot various Collatz chains.

plt.figure(figsize=(10,7))
for i in range(0,len(startingN)):
  X = range(0,startingN[i][1])
  Y = plottingN[i][0]
  plt.plot(X, Y, marker='.', color='red')

X = range(40,111)
Y = [14000*(3/4)**(x/3) for x in X]
plt.plot(X, Y, marker='', color='blue', label='k1=14000')
X = range(48,111)
Y = [29000*(3/4)**(x/3) for x in X]
plt.plot(X, Y, marker='', color='blue', label='k1=30000')
X = range(53,111)
Y = [50000*(3/4)**(x/3) for x in X]
plt.plot(X, Y, marker='', color='blue', label='k1=50000')
X = range(60,111)
Y = [97000*(3/4)**(x/3) for x in X]
plt.plot(X, Y, marker='', color='blue', label='k1=97000')

X = range(1,63)
Y = [0.00085*(3/2)**(x/2) for x in X]
plt.plot(X, Y, marker='', color='blue', label='k2=0.00085')
X = range(1,68)
Y = [0.0003*(3/2)**(x/2) for x in X]
```

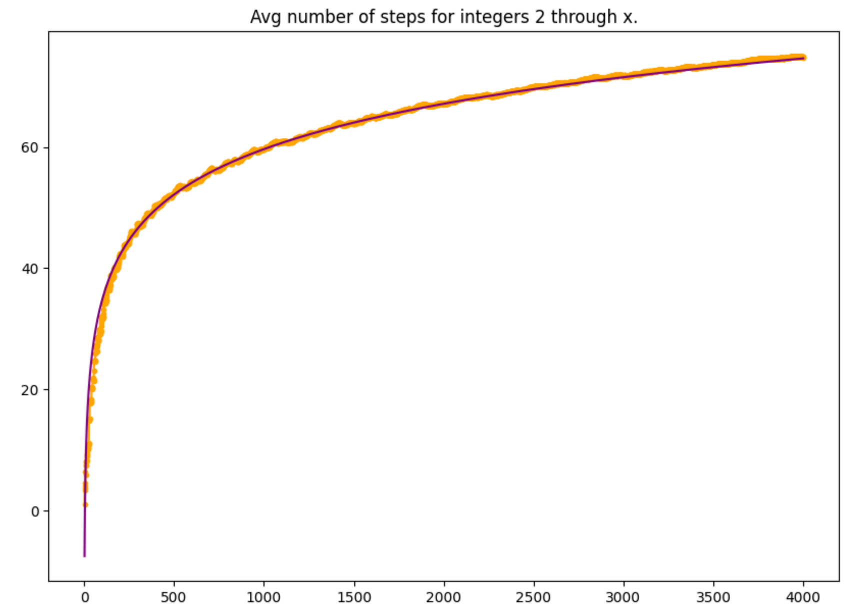


```
plt.plot(X, Y, marker='', color='blue', label='k2=0.0003')
X = range(1,73)
Y = [0.00012*(3/2)**(x/2) for x in X]
plt.plot(X, Y, marker='', color='blue', label='k2=0.00012')
X = range(1,77)
Y = [0.00005*(3/2)**(x/2) for x in X]
plt.plot(X, Y, marker='', color='blue', label='k2=0.00005')

plt.title("Plot of Collatz chains and bounding curves (k,b constant).")
plt.legend()
plt.show()
```



Next, we expand on previous ideas by computing the cumulative average number of the number of steps (i.e. chain length) for the starting integer 2 through n— then graph this value as a function of n. This resulting graph is logarithmic.

```
def stepList(n): # returns list of duples with [input value, number of steps].
  list = []
  for i in range(2,n+1):
    initial = i
    steps = 0
    while i != 1:
      i = collatz(i)
      steps += 1
    list.append([initial, steps])
  return list

X = []
Y = []
thisList = stepList(4000)
sumSteps = 0
for i in thisList:
  X.append(i[0])
  sumSteps += i[1]
  Y.append(sumSteps/(i[0]-1))
plt.figure(figsize=(10,7))
plt.plot(X, Y, marker='.', color='orange')
plt.plot(X, [10.8*sympy.log(x/4) for x in X], marker='', color='purple')
plt.title("Avg number of steps for integers 2 through x.")
plt.show()
```

To see how we obtained the specific approximating function that we did, consider a modified Collatz function $S(n)$ defined by:

$S(n) = \frac{n}{2}$, if n even; $\frac{3n+1}{2}$, if n odd.

Further, define $s(n)$ to represent the expected number of steps we take when applying $S(n)$ (either 1 step if even, or 2 steps if odd). Based on our previous arguments,

$s(n) = \frac{1}{2}(1 + s(\frac{n}{2})) + \frac{1}{2}(2 + s(\frac{3n+1}{2}))$.

Assuming that $s(n)$ increases logarithmically with n, i.e. $s(n) = k\ln(n)$,

$k\ln(n) = \frac{1}{2}(1 + k\ln(\frac{n}{2})) + \frac{1}{2}(2 + k\ln(\frac{3n+1}{2}))$.

Solving for k,

$k = \frac{3}{\ln(\frac{4}{3+1/n})}$,

Which goes to $\frac{3}{\ln(4/3)}$ for large n. Subtracting an additional 4 to account for error, $s(n) = \frac{3}{\ln(4/3)}\ln(n) - 4$.

Now the graph above is the average value of $s(n)$ for values less than or equal to n. Thus we take the average value, using integration, of $s(n)$:

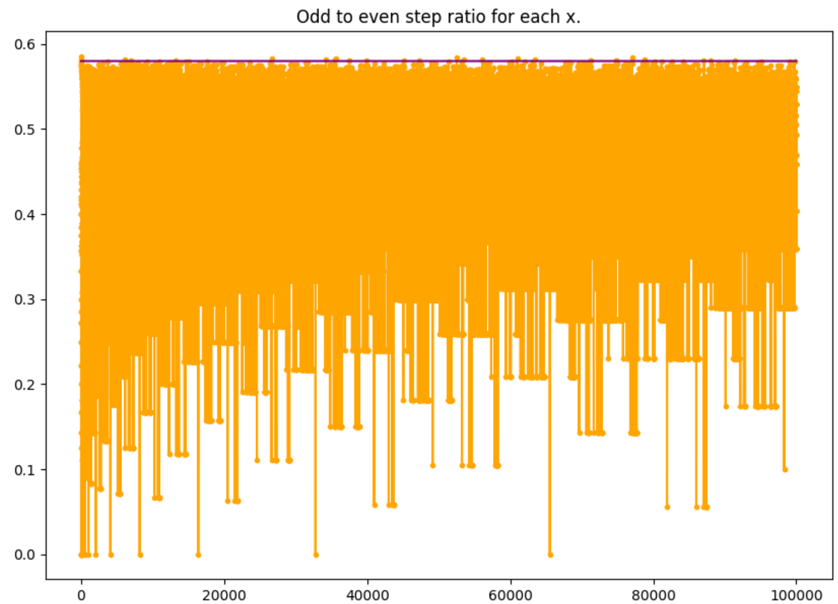
$\frac{1}{n}\int_0^n(\frac{3}{\ln(4/3)}\ln(t) - 4)dt = \frac{3}{\ln(4/3)}\ln(x) - 4 - \frac{3}{\ln(4/3)} \approx 10.5\ln(\frac{x}{4})$.

Let's try another thing. Instead of counting the number of total steps, we'll graph the *ratio* of the number of times the "$3n+1$" step occurs to the number of times the "$n/2$" step occurs. We should expect this ratio to approach 0.5 (a one to two ratio), as illustrated by our second graph (which informs us how many times we can divide by two before doing $3n+1$).

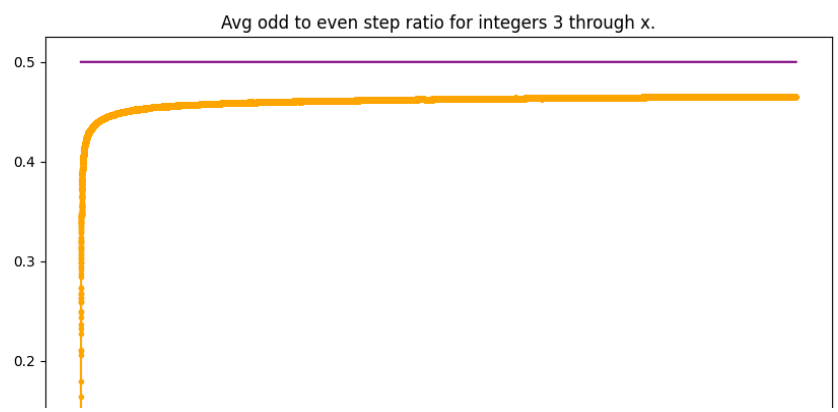
```
def stepList(n): # returns list of tuples with [input value, num of odd steps, num o
  list = []
  for i in range(2,n+1):
    initial = i
    evenSteps = 0
    oddSteps = 0
    while i != 1:
      if i%2 == 0:
        evenSteps += 1
      elif i%2 == 1:
        oddSteps += 1
      i = collatz(i)
    list.append([initial, oddSteps, evenSteps])
  return list

X = []
Y = []
thisList = stepList(100003)
for i in thisList:
  X.append(i[0])
  sumStepRatio = i[1]/i[2]
  Y.append(sumStepRatio) # look at ratio for each x

plt.figure(figsize=(10,7))
plt.plot(X, Y, marker='.', color='orange')
plt.plot(X, [0.58 for x in X], marker='', color='purple')
plt.title("Odd to even step ratio for each x.")
plt.show()

X = []
Y = []
thisList = stepList(100003)
sumStepRatio = 0
for i in thisList:
```

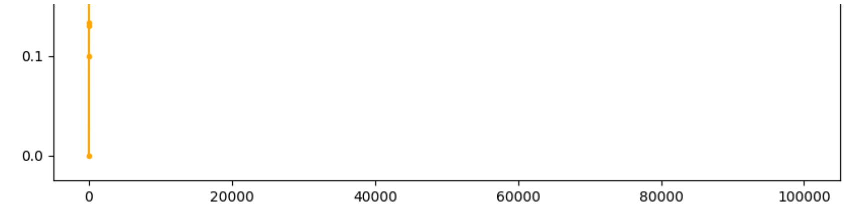


```
  X.append(i[0])
  sumStepRatio += i[1]/i[2]
  Y.append(sumStepRatio/(i[0])) # look at avg ratio for numbers 3 through x

print(X[10], Y[10])
print(X[32], Y[32])
print(X[100], Y[100])
print(X[316], Y[316])
print(X[1000], Y[1000])
print(X[3162], Y[3162])
print(X[10000], Y[10000])
print(X[31623], Y[31623])
print(X[100000], Y[100000])

plt.figure(figsize=(10,7))
plt.plot(X, Y, marker='.', color='orange')
plt.plot(X, [0.5 for x in X], marker='', color='purple')
plt.title("Avg odd to even step ratio for integers 3 through x.")
plt.show()
```




```
12 0.23209429459429462
34 0.29878987769132637
102 0.3559410387937496
318 0.4032545335732302
1002 0.4273256037258179
3164 0.443500238022575
10002 0.45465244887622236
31625 0.4607603662176044
100002 0.4654689524019759
```





Our lower bound is 0, unsurprisingly (as this occurs whenever n is a power of 2). But there appears to be an upper bound of ~0.58, instead of e.g. 1, as one migh initially expect— given the fact that the maximum rate at which n could increase (50% every two steps) occurs when the ratio of $n/2$ to $3n+1$ steps is 1:1.

To explain why this is the case, first recall that all integers here *eventually* reach one, so it is impossible for the chain to *always* be zig-zagging upward at the maximum rate— it clearly must come down eventually.

We can in fact account for this restriction by observing the fact that if the chain were to hold steady (neither go up nor down in the long term), there is be some fixed ratio at which the odd and even steps would iterate. Denote this ratio by a/b: then it must be the case that 3^a times $(1/2)^b$ is 1. So:

$3^a \cdot (1/2)^b = 1$

$3^a = 2^b$

$a \log 3 = b \log 2$

$a/b = \frac{\log 2}{\log 3} \approx 0.631.$

Hence, the ratio is bounded above by this number.

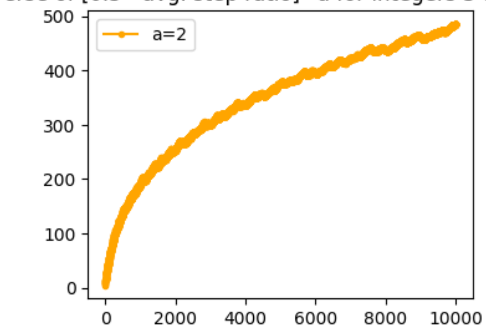
Now moving on to the second graph: it appears that the cumulative average ratio approaches 0.5 (indicating there are twice as many even steps, just as we had expected), but extremely slowly. To get a grip of how slowly this value approaches 1, we look at the plots of $1/(0.5-r)^a$, where r is the cumulative average ratio and a is some small integer.

```
X = []
Y = []
thisList = stepList(10003)
sumStepRatio = 0
for i in thisList:
  X.append(i[0])
  sumStepRatio += i[1]/i[2]
  Y.append(sumStepRatio/(i[0]))

plt.figure(figsize=(4, 3))
plt.plot(X, [-(sympy.log(0.5-y)+3)/(0.5-y) for y in Y], marker='.', color='orange',
plt.title("Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.")
plt.legend()
plt.show()

plt.figure(figsize=(4, 3))
plt.plot(X, [1/(0.5-y)**2 for y in Y], marker='.', color='orange', label='a=2')
plt.title("Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.")
plt.legend()
plt.show()

plt.figure(figsize=(4, 3))
plt.plot(X, [1/(0.5-y)**3 for y in Y], marker='.', color='orange', label='a=3')
plt.title("Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.")
plt.legend()
plt.show()

plt.figure(figsize=(4, 3))
plt.plot(X, [1/(0.5-y)**4 for y in Y], marker='.', color='orange', label='a=4')
plt.title("Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.")
plt.legend()
plt.show()

plt.figure(figsize=(4, 3))
plt.plot(X, [1/(0.5-y)**5 for y in Y], marker='.', color='red', label='a=5')
plt.title("Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.")
plt.legend()
plt.show()

plt.figure(figsize=(4, 3))
plt.plot(X, [1/(0.5-y)**6 for y in Y], marker='.', color='orange', label='a=6')
plt.title("Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.")
plt.legend()
plt.show()

plt.figure(figsize=(4, 3))
plt.plot(X, [1/(0.5-y)**7 for y in Y], marker='.', color='orange', label='a=7')
```

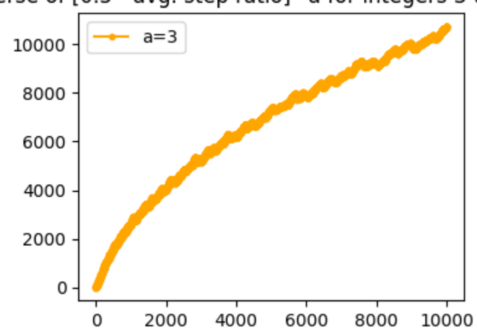


```
plt.title("Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.")
plt.legend()
plt.show()
```

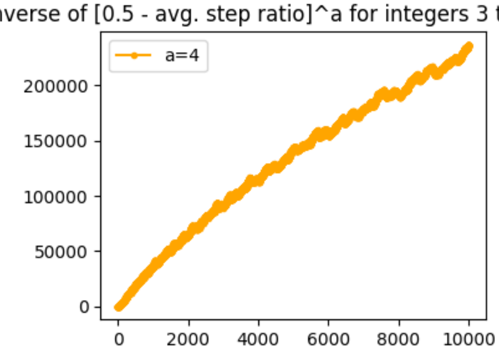
Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.



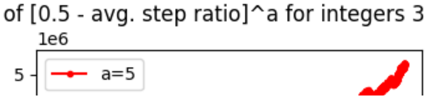
Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.

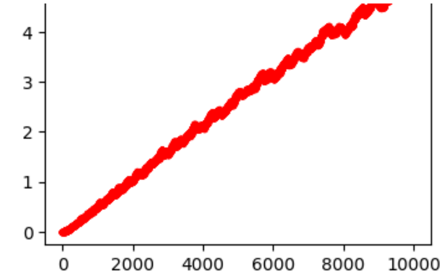


Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.

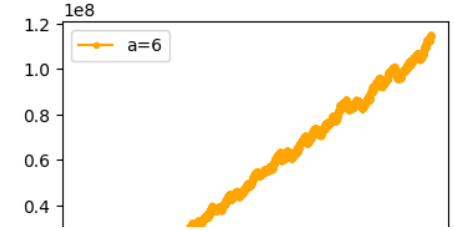


Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.





Inverse of [0.5 - avg. step ratio]^a for integers 3 through x.



Notice that while nearly all graphs are not linear, the graph of $(0.5 - r)^{-5}$ vs n, where r is again the average rato of odd to even steps via Collatz for integers 3 through n, is linear— that is,

$(0.5 - \frac{1}{n} \sum_{i=3}^{n} \frac{\text{odd steps for i}}{\text{even steps for i}})^{-5} \approx \frac{n}{2000}$.

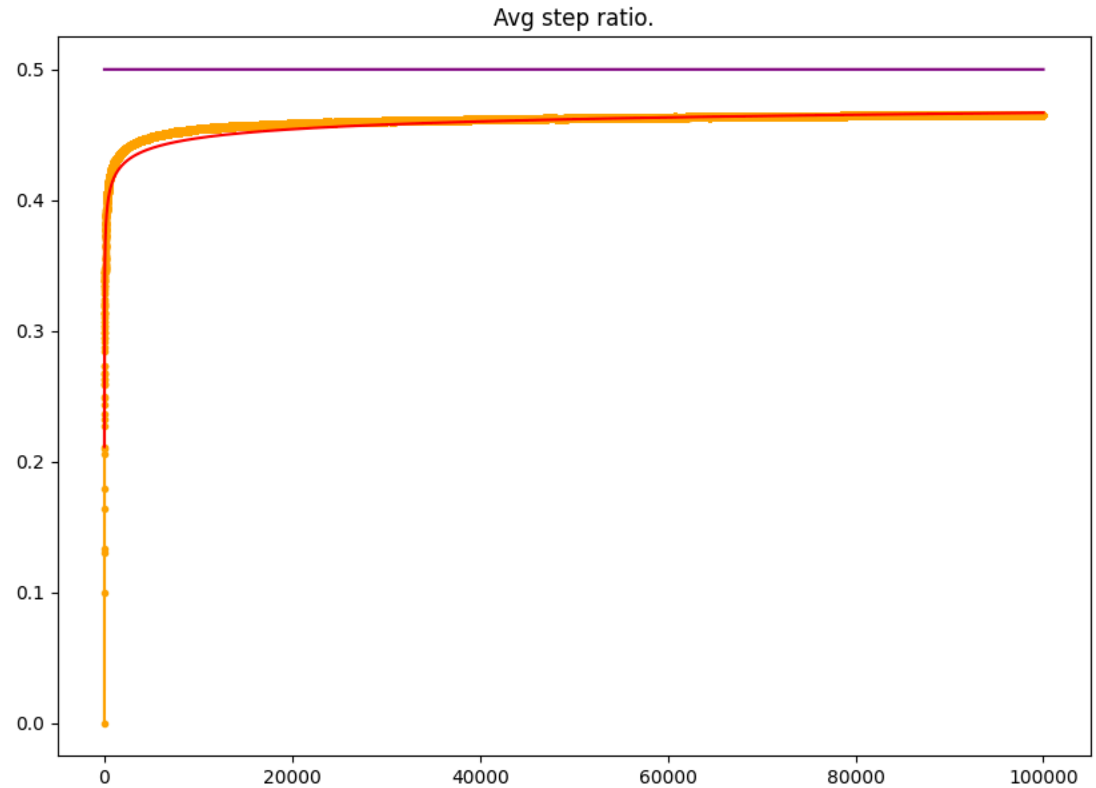
In other words,

$\frac{1}{n} \sum_{i=3}^{n} \frac{\text{odd steps for i}}{\text{even steps for i}} \approx 0.5 - (\frac{n}{2000})^{-\frac{1}{5}}$.

$r \approx 0.5 - (\frac{2000}{n})^{\frac{1}{5}}$.

Let's plot this as a function to check our approximation:

```
[ ] X = []
    Y = []
    thisList = stepList(100003)
    sumStepRatio = 0
    for i in thisList:
      X.append(i[0])
      sumStepRatio += i[1]/i[2]
      Y.append(sumStepRatio/(i[0])) # avg ratio for numbers 3 through x

    plt.figure(figsize=(10,7))
    plt.plot(X, Y, marker='.', color='orange')
    plt.plot(X, [0.5 for x in X], marker='', color='purple')
    plt.plot(X, [0.5-math.pow(0.004/x,0.2) for x in X], marker='', color='red')
    plt.title("Avg step ratio.")
    plt.show()
```



Since showing whether or note the growth of this average value indeed matches our function above may be very difficult, we pose this as a conjecture. In other words:

Is it true that the average ratio of odd to even steps for integers up to n is big O of $-n^{1/5}$?

This problem remains to be answered.

References.

The Notorious Collatz Conjecture (https://terrytao.wordpress.com/wp-content/uploads/2020/02/collatz.pdf). This presentation by Terrence Tao may be a good starting point in the investigation of basic properties of the Collatz conjecture.

Specifying and Verifying the Convergence Stairs of the Collatz Program (https://arxiv.org/pdf/2403.04777). This article explains the "invariant" of the Collatz problem— i.e. given a number, which numbers lead to it via Collatz? This concept is important in creating "Collatz trees."

Elementary Number Theory, A Computational Approach (https://wstein.org/edu/2007/spring/ent/ent.pdf). This book by William Stein will be a good source of number theory background.

The 3x+1 problem: An annotated bibliography. 1963-1999: (https://arxiv.org/abs/math/0309224); 2000-2009: (https://arxiv.org/abs/math/0608208).