FiveThirtyEight's August 7, 2020 Riddler

Emma Knight

August 11, 2020

This weeks riddler, courtesy of Kareem Carr, is as follows:

Question 1. *We usually think of addition as an operation applied to a field like the integers or the real numbers. And there is good reason for that - as Kareem says, "Mathematicians have done all the hard work of figuring out how to make calculations track with reality. They kept modifying and refining the number system until everything worked out. It took centuries of brilliant minds to do this!"*

Now suppose we defined addition another (admittedly less useful) way, using a classic model organism: the nematode. To compute the sum of x and y, you combine groups of x and y nematodes and leave them for 24 hours. When you come back, you count up how many you have - and that's the sum!

It turns out that, over the course of 24 hours, the nematodes pair up, and each pair has one offspring $1/2$ of the time. If you have an odd number of nematodes, they will still pair up, but one will be left out. So if you want to compute $1+1$, half the time you'll get 2 and half the time you'll get 3. If you compute $2+2$, $1/4$ of the time you get 4, $1/2$ of the time you'll get 5, and $1/4$ of the time you'll get 6.

While we're at it, let's define exponentiation for sums of nematodes. Raising a sum to a power means leaving that sum of nematodes for the number of days specified by the exponent. Importantly, all the nemotodes re-pair after every day.

With this number system, what is the expected value of $(1+1)^4$?

Extra credit: As N gets larger and larger, what does the expected value of $(1+1)^N$ approach?

It's easy to start iterating by hand:

	2	3	4	5	6	7	8	9
0	1							
1	$\frac{1}{2}$	$\frac{1}{2}$						
2	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$					
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{5}{16}$	$\frac{1}{8}$	$\frac{1}{16}$			
4	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{17}{64}$	$\frac{3}{16}$	$\frac{19}{128}$	$\frac{7}{128}$	$\frac{3}{128}$	$\frac{1}{128}$

This table shows the probability of there being n frogs on day d (I am adopting the convention that blank entries are zeroes). This is basically the limit of what I can do by hand, but it is enough to get the answer to the main question on the riddler. The expected number of nemotodes on day 4 is $\frac{141}{32} = 4.40625$.

Before doing any computations for N large, I will first give a heuristic. If the number of nemotodes on day k is n, then there are $\lfloor \frac{n}{2} \rfloor$ pairs of nemotodes that can produce offspring, and so, on average, there are $n + \frac{\lfloor n/2 \rfloor}{2}$ new nemotodes. If $n \gg 0$, then this is roughly $\frac{5n}{4}$. Applying linearity of expectation a bajazillion times, one gets that, for $N \gg 0$, $\mathbb{E}\left((1+1)^{N+1}\right) \approx \left(\frac{5}{4}\right) \mathbb{E}\left((1+1)^N\right)$. Thus, one expects that, for N large, $\mathbb{E}\left((1+1)^N\right) = C\left(\frac{5}{4}\right)^N$ where C is some constant that incorporates how long it takes for N to actually get sufficiently large.

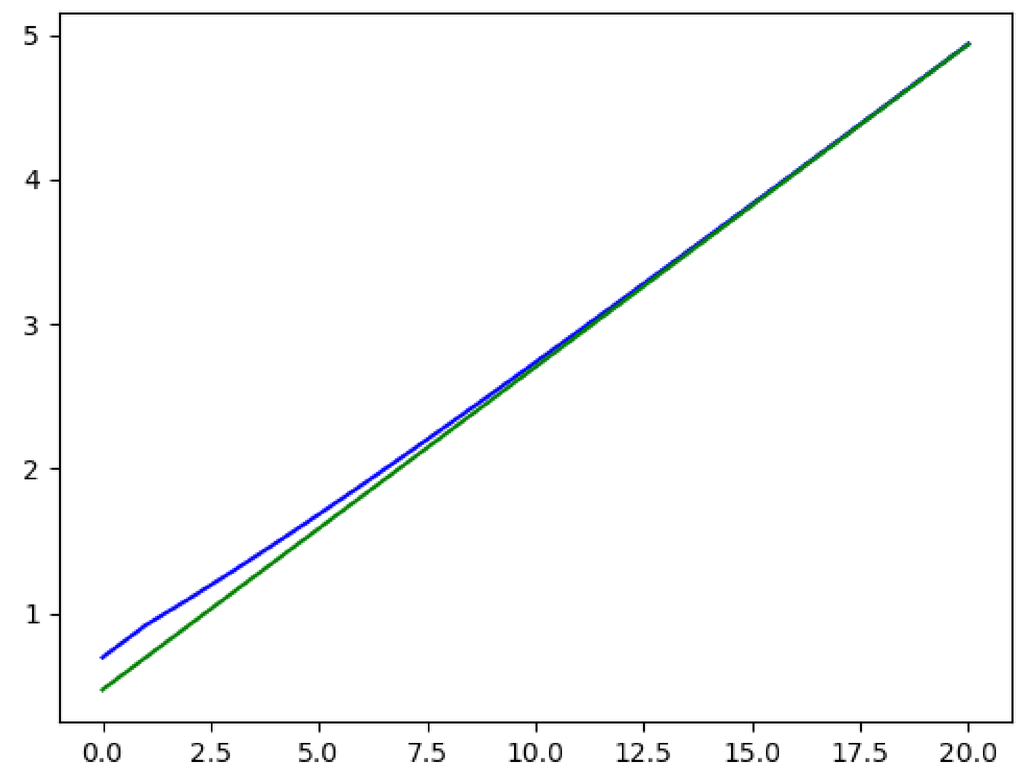
One can dress this up a little bit to reach the same conclusion: if there are n nemotodes at time N, then there are at least $\frac{5n-1}{4} = \frac{5n}{4}\left(1 - \frac{1}{5n}\right)$ nemotodes on average at time $N+1$. Thus, writing $x_N = \mathbb{E}\left((1+1)^N\right)$ and $y_N = \log(x_N)$, one has that $y_{N+1} \approx y_N + \log(5/4) + \log\left(1 - \frac{1}{5x_N}\right) \approx y_N + \log(5/4) - \frac{1}{5x_N}$, with the understanding that this is probably an underestimate. Now, one starts to apply this k times and sees that $y_{N+k} \approx y_N + k\log(5/4) - \frac{1}{5}\sum_{i=0}^{k-1} \frac{1}{x_{N+i}}$. Since the x_Ns are growing exponentially, one has that that sum converges. Thus, one should get that $y_{N+k} > y_N + k\log(5/4) - C$ where C is some absolute constant. It is also pretty clear that $y_{N+k} \leq k\log(5/4) + y_N$. There is a lot of delicacy to make this precise and while I'm not 100% sure it can be done, it feels like it should be possible. The upshot is that $y_N = N\log(5/4) + O(1)$ and so $x_N = \Theta\left(\left(\frac{5}{4}\right)^N\right)$. However, it is also clear that $\frac{x_{N+1}}{x_N} < \frac{5}{4}$, so the sequence $\frac{x_N}{(5/4)^N}$ is monotonically decreasing and bounded away from 0, so one gets that there is a constant C such that $x_N \sim C\left(\frac{5}{4}\right)^N$.

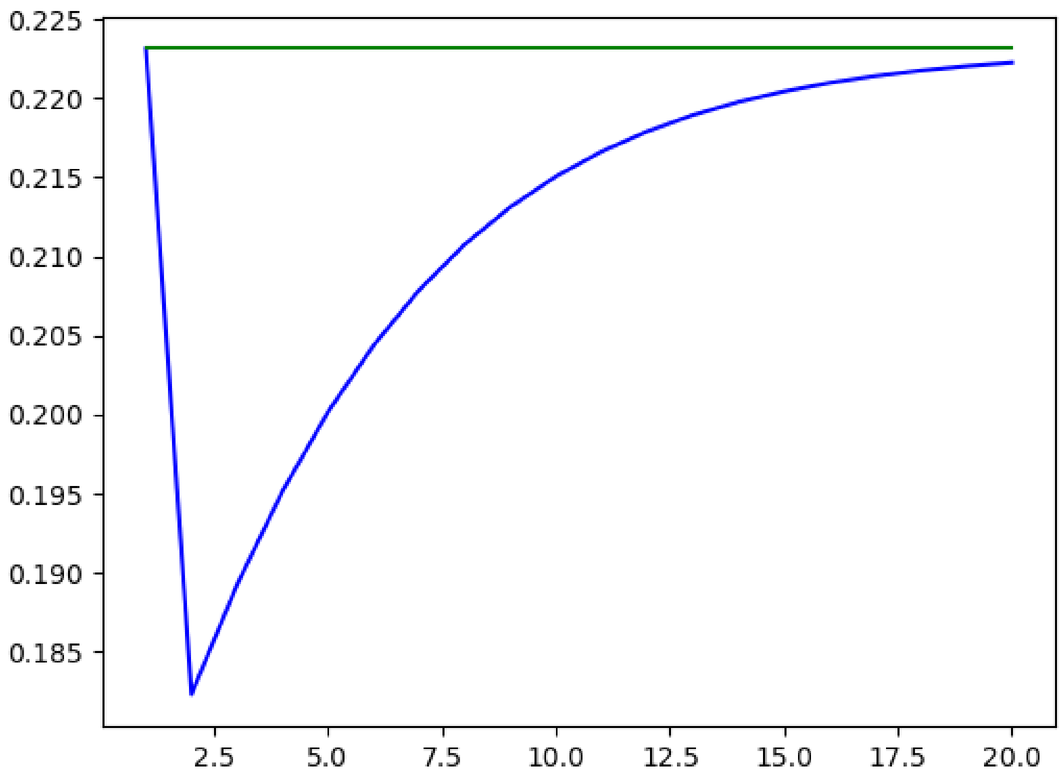
And now, to the python!

```
import math
import matplotlib.pyplot as plt

##This code starts off with an implementation of the
##dyadic rationals, where a/2^n is represented as [a, n].
##dy(a) takes in an integer and returns it as a dyadic
##rational.  plus and times add and multiply two dyadic
##rationals.  simp simplifies a dyadic rational, removing
```

```
##superfluous powers of two from the denominator.  value
##spits out the value of the dyadic rational as a float.

def dy(a):
    return [a, 0]

def plus(d1, d2):
    a1 = d1[0]
    a2 = d2[0]
    n1 = d1[1]
    n2 = d2[1]
    if (n1 >= n2):
        n = n1
        a = a1 + (2**(n1-n2))*a2
    if (n2 > n1):
        n = n2
        a = a2 + (2**(n2-n1))*a1
    return([a, n])

def times(d1, d2):
    a1 = d1[0]
    a2 = d2[0]
    n1 = d1[1]
    n2 = d2[1]
    return([a1*a2, n1+n2])

def simp(d):
    a = d[0]
    n = d[1]
    while ((a % 2 == 0) and (n > 0)):
        a = a//2
        n = n-1
    return([a, n])

def value(d):
    d = simp(d)
    a = d[0]
    n = d[1]
    return(a*((.5)**(n)))

##This is the core loop.  This takes in a list of dyadic rationals,
##views the kth entry as the probability that there are k nemotodes
##at the start, and lets the nemotodes reproduce for n days.

def iterate(l, n = 1):
    if (n == 0):
        return(l)
    new = []
    for i in range(len(l)):
        k = i//2
        for j in range(k+1):
            if (i+j < len(new)):
                new[i+j] = plus(new[i+j], times(l[i], [math.comb(k, j), k]))
            if (i+j == len(new)):
                new.append(times(l[i],[math.comb(k, j), k]))
    return(iterate(new, n-1))

##This displays a list as above in a more human readable
##format.

def display(l):
    for i in range(len(l)):
        print(i, simp(l[i]), value(l[i]))

##This computes the expected number of nemotodes on a list.

def total(l):
    s = [0, 0]
    for i in range(len(l)):
        s = plus(times(dy(i), l[i]), s)
    return(simp(s))

##This is the main computational loop.  I will start with two
##nemotodes with probability 1, and run for n days.  The arrays
##that I define are for graphing, with results being the natural
##log of the results, guess1 being the predicted guess for the
##results, deltas being the deltas of results, and guess2 being
##the predicted deltas of the results.  I start with this list,
##iterate it, append the log of the total of the new list, and
##repeat.  The program will then show the graphs of results vs.
##guess1, and then deltas vs. guess2.  Guesses will be in green,
##and actual results will be in blue.  Finally, the commented
##out code shows you what the value is after i iterations and how
##it compares to (5/4)^i.

l = [[0,0], [0, 0], [1, 0]]
n = 20
results = [math.log(2)]
guess1 = [math.log((8/5))]

for i in range(n):
    l = iterate(l)
    d = total(l)
    results.append(math.log(d[0]) - d[1]*math.log(2))
    guess1.append(math.log(2) + (i * math.log(5/4)))
##    v = value(d)
##    print(2*((1.25)**(i+1)), v, v/(2*((1.25)**(i+1))))

deltas = []
guess2 = []
for i in range(1, len(results)):
    deltas.append(results[i]-results[i-1])
    guess2.append(math.log(1.25))

plt.plot(range(n+1), results, 'b-')
plt.plot(range(n+1), guess1, 'g-')
plt.show()
plt.cla()
plt.plot(range(1, n+1), deltas, 'b-')
plt.plot(range(1, n+1), guess2, 'g-')
plt.show()
```

The code suggested that $x_N \sim \left(\frac{8}{5}\right)\left(\frac{5}{4}\right)^N$, with the values getting close at $N = 20$. Here are the two graphs that the code produced:





It is easy to see that the guesses become very good by $N = 20$. This was about the limit of what my computer could handle, so I think that $x_N \sim \left(\frac{8}{5}\right)\left(\frac{5}{4}\right)^N$ seems like a reasonable guess but one would need to go further to have some amount of certainty.

Addendum: After seeing some other solutions, there is indeed a closed form for x_N for all N. Notice that, for $N \geq 1$, the probability that there are an odd number of nemotodes is $\frac{1}{2}$ (fix a total number of nemotodes after N days, fix one pair of nemotodes, and condition on how many of the other pairs of nemotodes have offspring. Then the odds that this pair makes the total number of nemotodes odd on day N is $\frac{1}{2}$ no matter what, so summing up over all the possible number of nemotodes on day N and all the possibilities of offspring from other nemotodes, one still gets $\frac{1}{2}$ for the odds that there are an odd number of nemotodes on day $N+1$). Additionally, if there are k nemotodes after N days, one easily sees that, with k even, the expected number of nemotodes is $\frac{5k}{4}$ and if k is odd the number is $\frac{5k}{4} - \frac{1}{4}$. Summing this up over all the possibilities, one gets that $x_{N+1} = \frac{5x_N}{4} - \frac{1}{8}$. Letting $x'_N = x_N - \frac{1}{2}$, one has that $x'_{N+1} = x_{N+1} - \frac{1}{2} = \frac{5x_N}{4} - \frac{5}{8} = \frac{5}{4}\left(x_N - \frac{1}{2}\right) = \frac{5x'_N}{4}$. Thus, since $x'_1 = 2$, one has that $x_N = \left(\frac{8}{5}\right)\left(\frac{5}{4}\right)^N + \frac{1}{2}$ for $N \geq 1$.