FiveThirtyEight's August 19, 2022 Riddler

Emma Knight

August 21, 2022

This week's riddler, courtesy of Dave Moran, is about chugging perscription drugs:

Question 1. *I've been prescribed to take* 1.5 *pills of a certain medication every day for* 10 *days, so I have a bottle with* 15 *pills. Each morning, I take two pills out of the bottle at random.*

On the first morning, these are guaranteed to be two full pills. I consume one of them, split the other in half using a precision blade, consume half of that second pill, and place the remaining half back into the bottle.

On subsequent mornings when I take out two pills, there are three possibilities:

- *I get two full pills. As on the first morning, I split one and place the unused half back into the bottle.*
- *I get one full pill and one half-pill, both of which I consume.*
- *I get two half-pills. In this case, I take out another pill at random. If it's a half-pill, then I consume all three halves. But if it's a full pill, I split it and place the unused half back in the bottle.*

Assume that each pill — whether it is a full pill or a half-pill — is equally likely to be taken out of the bottle.

On the 10th day, I again take out two pills and consume them. In a rush, I immediately throw the bottle in the trash before bothering to check whether I had just consumed full pills or half-pills. What's the probability that I took the full dosage, meaning I don't have to dig through the trash for a remaining half-pill?

The way to do this is to think backwards: on the last day, you must have either 3 half pills or 1 full pill and 1 half pill. If you have 3 half pills you will need to dig in the trash, and if you have 1 of each you won't.

On the penultimate day, you might have 6 half pills, 4 half pills and 1 full pill, 2 of each, or 3 full pills. You can compute the odds that you need to dig things out of the trash in each circumstance (for example, with 2 each, you have a one in six chance of ending up with 3 half pills). You can repeat this for each day, and there are closed formulae for the odds of moving to the various states that I'm not going to write down because of how messy they are.

This leads you to want to write code to compute these odds recursively (which will be shown at the end of this document). This gives a probability of needing to dig through the trash as roughly 20.988%. One interesting thing that you might notice is that, as the number of pills you take goes up, the odds of needing to dig in the trash goes up, reaching 51.156% at 1500 pills. One way to think about this: assume you started with an enormous number of pills, and an enormous amount of time has passed but you still have a large number of pills and half pills left. Assume you have a full pills and b half pills, and let the density of full pills be x (which is just $\frac{a}{a+b}$. The expected change in density is very close to $(x^2(x-2) + x(1-x)(2-x)(2x-1) + 3x(1-x)^3)/(a+b)$ (the three terms correspond to going from (a, b) to $(a-2, b+1)$, $(a-1, b-1)$, and $(a, b-3)$ respectively and computing what it does to the change in density). Setting the polynomial above equal to 0, one gets that $x \approx .31767$. Thus, in the stable state, you have more than twice as many half pills as full pills. Additionally, if you get unlucky and run out of full pills, they aren't coming back whereas if you get lucky and run out of half pills, they can still come back. Thus, you are more likely to wind up hitting the "no full pills" situation than the "no half pills" and there is no recovery from getting there.

Anyways, here's the code:

```
tot = 1000
queue = []
for n in range(1, tot+1):
    for a in range(1+(3*n)//2):
        b = 3*n-2*a
        queue.append([a, b])
ys = ['a']*(3*tot+1)
values = []
for i in range(2*tot+1):
    values.append(ys.copy())

for p in queue:
    x = p[0]
    y = p[1]
    if p == [0, 3]:
        values[x][y] = 1
    elif p == [1, 1]:
        values[x][y] = 0
    elif x == 0:
        values[x][y] = values[x][y-3]
    elif y == 0:
        values[x][y] = values[x-2][y+1]
    elif y == 1:
        values[x][y] = (2*values[x-1][y-1] + (x-1)*values[x-2][y+1])/(x+1)
    elif y == 2:
        values[x][y] = ((4*x+2)*values[x-1][y-1] + (x)*(x-1)*values[x-2][y+1])/((x+2)*(x+1))
    elif x == 1:
        values[x][y] = ((y-2)*values[x][y-3] + 3*values[x-1][y-1])/(y+1)
    else:
        values[x][y] = (y*(y-1)*(y-2)*values[x][y-3] + x*y*(2*x+3*y-5)*values[x-1][y-1]
                        + x*(x-1)*(x+y-2)*values[x-2][y+1])/((x+y)*(x+y-1)*(x+y-2))

for k in range(tot//2):
    print(3*k+3, values[3*k+3][0])
```