Slow Car Chase

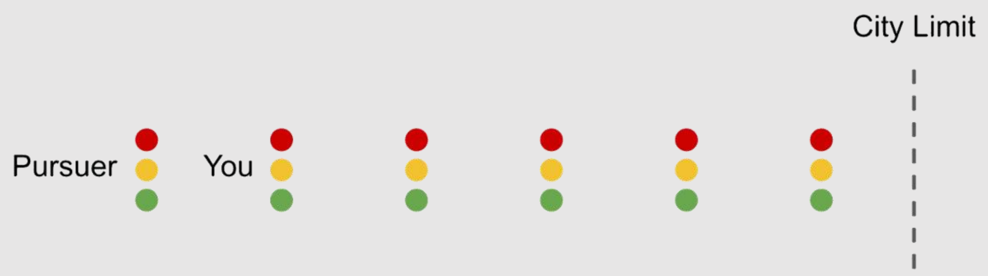
This week's Fiddler on the Proof (20 October 2023) asks:

You find yourself engaged in a car chase, but it differs from the movies in one key way: Both you and the car that's chasing you obey all traffic laws, being sure to stop at any and all red lights.

You're trying to make it to the city limit, but you're currently one block ahead of your pursuer. Right now you're both at a red light. The light turns green at the same time for both of you, at which point both cars begin moving east at a speed of one block per minute.

Every time that you come to an intersection, there's a 50 percent chance the light is green, in which case you coast right on through. But there's also a 50 percent chance the light is red, in which case it takes exactly one minute for the light to turn green again. These same probabilities govern your pursuer—at each intersection, they have a 50 percent chance of encountering a green light and a 50 percent chance of encountering a red light and having to wait one minute, entirely independent of whatever you might have encountered at that same intersection.

Including the light at which you are now stopped, there are five traffic lights between you and the city limit, as illustrated below. That means there are six lights for your pursuer.



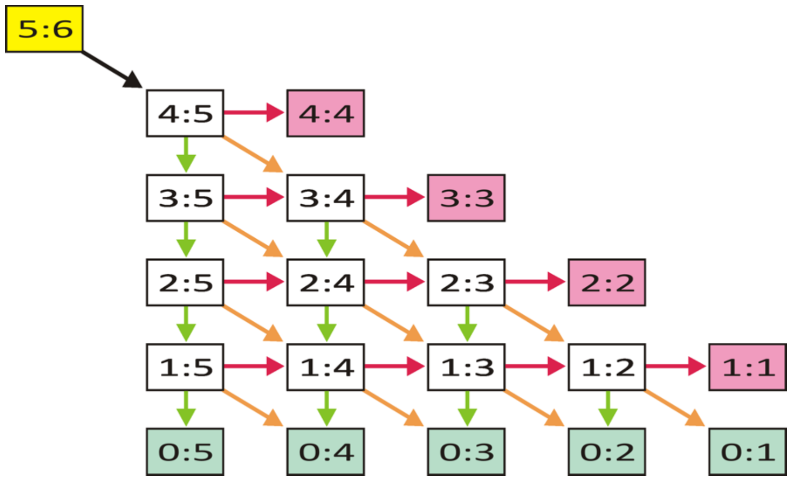
How likely is it that you'll make it past the city limit without being caught?

Mathematical Model

Often the hardest part of solving a puzzle is figuring out how to convert the English description into a model that can be tackled mathematically.

There are two critical pieces of information that change as the chase going on: (1) how many lights you have to pass through and (2) how many lights pursuer has to pass through. Knowing these two things tells you what "state" the chase is in.

Here is a chart of the possible states:



Each box in the chart represents a state in the chase and contains two numbers. The first number is how many lights **you** must pass through to escape the city. The second number is how many lights **your pursuer** must pass through.

The yellow box is the current state – you have 5 lights to go and pursuer has 6. The pink boxes are the states where pursuer caught up with you – his lights to go is the same as yours. The turquoise boxes are the states where you have successfully eluded pursuer – you have no more lights to go. The white boxes are those states where the pursuit is still on.

We are told that you and pursuer are both stopped at red lights and that the lights will turn green at the same time. So state "5:6" always changes to become state "4:5".

From any white state, four things can happen, with equal probability:

1. You and pursuer both arrive at a red light.
2. You arrive at a red light and pursuer arrives at a green light.
3. You arrive at a green light and pursuer arrives at a red light.
4. You and pursuer both arrive at a green light.

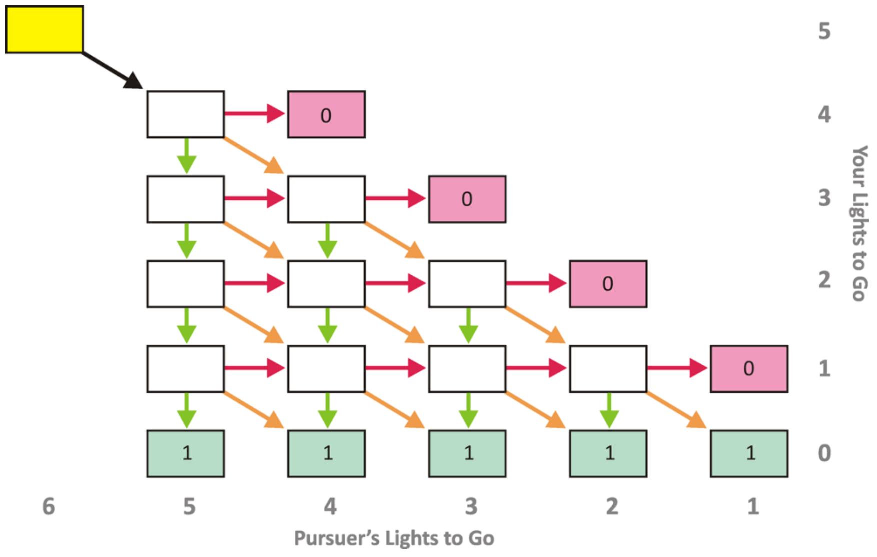
Cases 1 and 4 amount to the same thing: You and pursuer both reduce your lights-to-go by 1. (In case 4, this happens immedately. In case 1, you have to wait a minute first.) These cases (shown by yellow arrows in the chart) happen 50% of the time.

Case 2 is bad for you. It means pursuer gains on you. This case (shown by red arrows in the chart) happens 25% of the time.

Case 3 is good for you. It means you gain on pursuer. This case (shown by green arrows in the chart) happens 25% of the time.

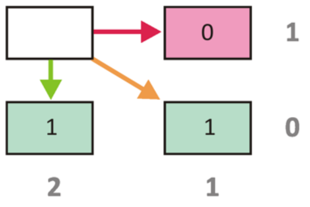
Probabilities

We now have enough information to calculate the probability of success from each of the various states. Let's start with the easy ones.



If you arrive at a pink state, it means you have been caught! Your chance of evasion is 0. If you arrive a turquoise state, it means you have escaped. Your chance of evasion is 1.

What about this other states? Those can be calculated one at a time starting at the lower right and moving backwards. Take state "1:2" as an example:

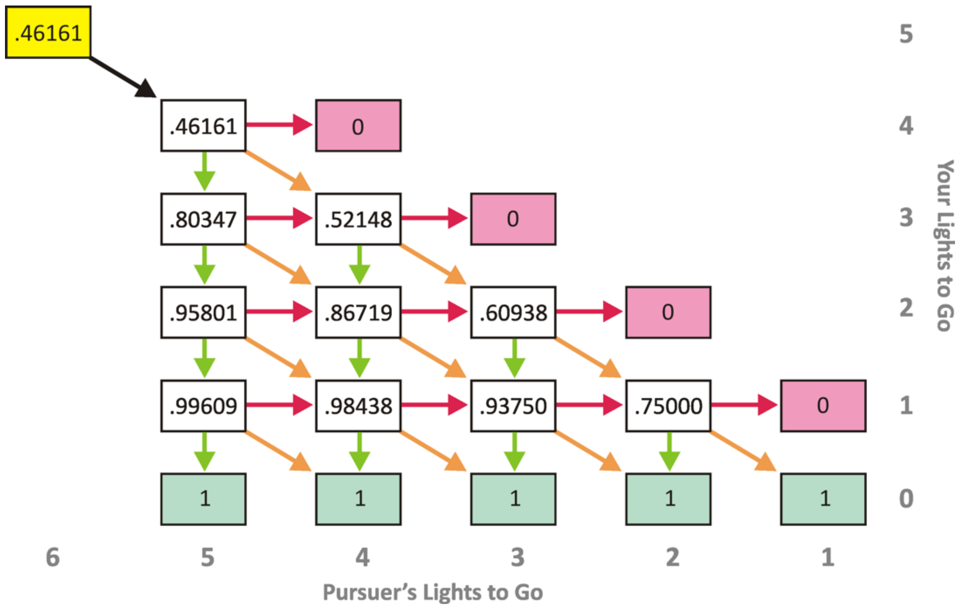


Three things can happen from here:

- 25% of the time you follow the green arrow and reach probability 1
- 50% of the time you follow the yellow arrow and reach probability 1
- 25% of the time you follow the red arrow and reach probability 0

$$.25 \times 1 + .5 \times 1 + .25 \times 0 = .75$$

The other states' probabilities are calculated the same way. Here are all the results:



Extra Credit

Now suppose Fiddler City is infinitely large. Once again, you find yourself stopped at a red light, heading due east, with your pursuer one block behind you. Both your light and your pursuer's light turn green the same instant, and all the other details (i.e., speeds, probabilities, and timings) are the same as before.

On average, how many minutes will it be until you are ultimately caught?

In one way, this is an easier problem than the main puzzle. We don't have to keep track of the distance-to-go to the city limits because there are no city limits. Theoretically, we can compute the average minutes until you're caught using this function:

```
func E( lead:Int ) -> Double
    {
    if lead == 0 { return 0.0 }
    return 1 + ( 0.25 * E( lead:lead-1 )
               + 0.50 * E( lead:lead )     ⚠ Function call causes an infinite recursion
               + 0.25 * E( lead:lead+1 ) )
    }
```

But the editor is showing an ominous warning: "Function causes an infinite recursion". We can't run this function because it will never return.

So instead what I did was create an array to use as backing storage for the function. Instead of having function E call itself, I had it use the value in the backing array instead.

What values should the backing array have? Initially, I just filled it with zeros. The plan was to compute the values of the array over and over again by calling function E until those values reached an equalibrium.

How big should the backing array be? I tried making it 50 entries; beyond a lead of 50, the function would use an abitrary value of zero. The hope was that the chance of reaching a lead of more than 50 was so small it wouldn't make a difference to the result.

After the values in the backing array stablized, I got the following result:

Size of backing array	E(lead:1)
50	100.0

To see how sensitive this result was to the size of backing array used, I tried larger arrays.

Size of backing array	E(lead:1)
100	200.0
150	300.0
200	400.0

There was an unsettling trend. The larger I made the backing array, the larger the value of E became – E seemed to grow without limit. So that's my answer:

The average number of minutes until you are caught is infinite.