CME241$_{a}ssignment2$

Quinn Hollister (bh9vw)

January 2022

1 Problem 1

Model the game of Snake and Ladders as a Markov Process. Write out it's state space and structure of transition probabilities. This is a slightly tricky state space, considering it could be thought of as just the numbering from 1 to 99 (excluding 100 because it is a terminal state). However, the snakes and ladders actually condense our state space since certain "states" aren't actually there. For example, there is a "ladder" going from 4 to 14, so if we go from 3 to 4, the transition is actually 3 to 14, and the state corresponding to 4 isn't a state at all. Thus our state space looks more like:

$$\mathcal{S} = \{38, 2, 3, 14, ..., 97, 78, 99\}$$

And our transition probabilities are defined as:

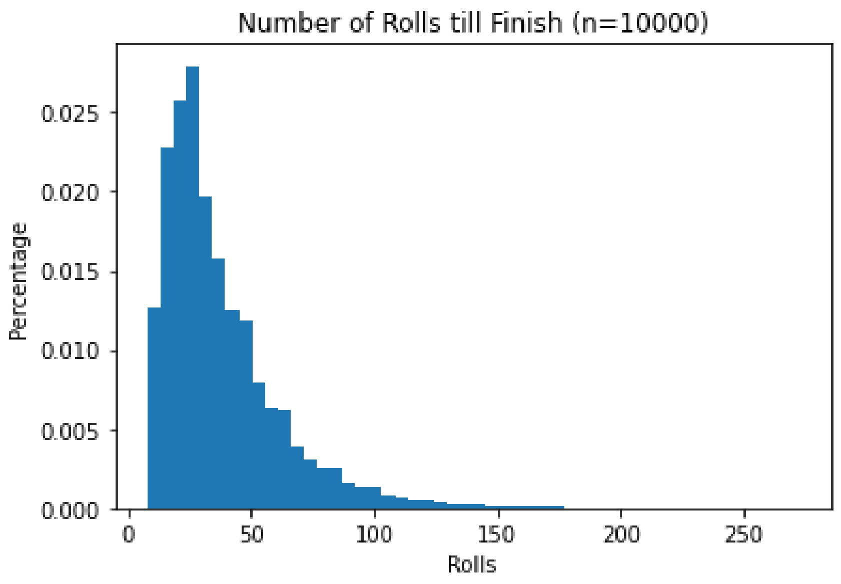
$$P(s+i|s) = 1/6, \forall i = \{1, 2, 3, 4, 5, 6\}$$

where the s, i variables refer to the index on the set $\mathcal{S}$, so $s+i = 3+1$ corresponds to the state 14. Also note that we start the game at 0, and end at 100, with no states after 100. This means that the transition probabilities "accumulate" for the terminal state 100. Thus, if we're at 99, then the transition probability to get to the terminal state is 1.

Just an FYI, I wasn't sure if I needed to fully express the entire state space with its intricacies in the latex paper, but just know its explicitly done in the code that follows.

2 Problem 2

In this problem, we are asked to model the game as an instance of a FiniteMarkovProcess. We then need to plot a graph of the probability distribution of time steps to finish the game. I created a class called **SnakesLadders** that implemented the transition map outlined in Problem 1, using the dataclass **StateSL**. Then, I ran 100,000 sampling traces and kept track of how long each trace took to get to the terminal state 100. The resulting histogram illustrates the distribution:



3 Problem 3

Solve the Frog Puzzle

We have a frog that has n-1 lilypads in front of it. It can jump to any of the lilypads or the bank with equal probability, what is the expected number of jumps to get to the bank? We can set this problem up with the following recursion, and note that my convention is we start at the 0th lilypad so that we have n-1 lilypads in front:

$$E[0] = \frac{1}{n} \cdot E[1] + ... + \frac{1}{n} \cdot E[n-1] + 1$$

But, this form works just as well for saying that we have the same problem as above, but now we're at the 1st lilypad:

$$E[1] = \frac{1}{n-1} \cdot E[2] + ... + \frac{1}{n-1} \cdot E[n-1] + 1$$

We can continue this until we get to the (n-1) lilypad, but we need an analytical formula for the 0th lilypad. The fact that we have the expectation for all of the other lilypads complicates finding an analytical formula, but we can use two scalars on our first and second expression to cancel out most of the intermediate terms.

To see this, multiply our first expression by n, and the second by (n-1), and then subtract:

$$n \cdot E[0] - (n-1) \cdot E[1] = E[1] + ... + E[n-1] + n - (E[2] + ... + E[n-1] + n - 1)$$

$$n \cdot E[0] - (n-1-n) \cdot E[1] = 1$$

$$E[0] = E[1] + 1/n$$

But, we can see that this just iterates until we get to the expression for the (n-1) lilypad, where $E[N] = 0$, so $E[N-1] = 1$. Thus, our expression becomes:

$$E[0] = 1 + ... + 1/n$$

$$E[0] = \sum_{i=1}^{n} 1/i$$

4 Problem 4

We now need to extend the setup of the Snakes and Ladders game from Problems 1 and 2 to include the Reward structure of a MRP. The rewards need to help us in finding a numerical solution for the expected dice rolls to finish the game from the starting position.

We can do this by associating each transition with the reward "1", which will essentially 'count' the number of rolls till finish. The idea is similar to the one we had in Problem 3, where our expected time to reach the bank was a function of the expected time for the other possible states we could jump to, normalized by the probability of the jump, plus a 1 to account for the fact that any of these jumps required a singular current "jump".

Thus, for each transition, we will associate a reward of 1 with it. By implementing a very similar class to our Problem 2 class, but extending the FiniteMarkovRewardProcess class, we were able to call the display value function, using a gamma of 1 (only since its a FiniteMarkovRewardProcess, and because we want each "jump" to have the same weight, regardless of when it occurs). This solved the LA equation corresponding to the transition and rewards map we implemented, and showed that at square 0, our expected number of rolls were 37.294.

We can have some faith in this result, as the mean "rolls till finish" of our numerical traces was 37.38, putting our relative error at 0.23 percent.