Assignment 2

Miao-Chin Yen

March 7, 2022

Problem 1 (Snakes and Ladders)

In this problem, we consider the game of Snakes and Ladders (single-player game) with 100 grids. Also, we assume that there are 6 ladders and 5 snakes. Let $Ladder = \{(1,38),(4,14),(9,31),(28,84),(80,100),(71,91)\}$ denote the taking-up route by ladders and $Snake = \{(93,73),(95,75),(87,24),(47,26),(17,6)\}$ denote the taking-down route by snakes where (i,j) stands for the transition from state i to state j. The Markov Process of this game is stated as follows:

(1) Let $\mathcal{S} = \{state : 1 \leq state \leq 100\}$ be the state space and $\mathcal{T} = \{100\}$ be the set of terminal states.

(2) Denote S_t as the state for time step t.

(3) Denote $\mathcal{P} : \mathcal{N} \times \mathcal{S} \rightarrow [0,1]$ as the Transition Probability Function (source $s \rightarrow$ destination s') where

$$\mathcal{P}(s,s') = \mathbb{P}[S_{t+1} = s' \mid S_t = s] \text{ for all } t = 0,1,2,\ldots$$

(4) Termination: If $S_T \in \mathcal{T}$, the process terminates at time step T.

For the specific 100 grids game, we can explicitly write the transition probability as follows:

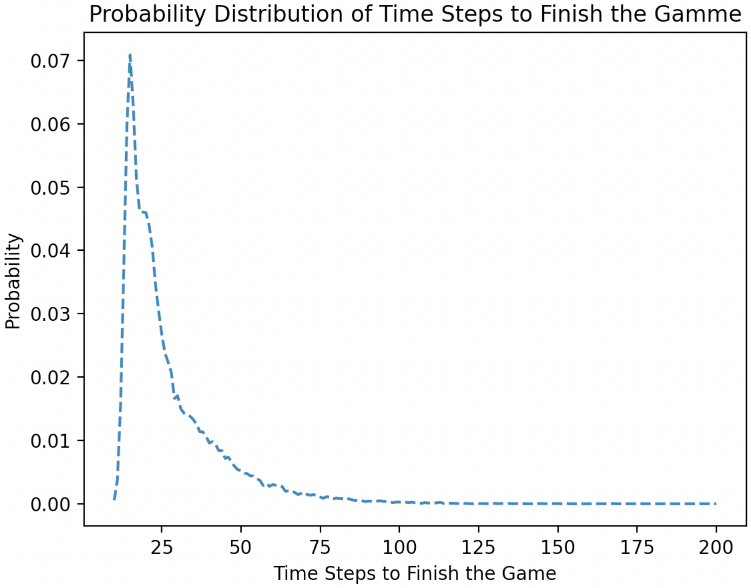
- $\forall (i,j) \in Ladder \bigcup Snake$:
$$\mathcal{P}(j,i) = 1$$
- $\forall i \notin$ the first component of $Ladder \bigcup Snake$, if $i + k < 100$
$$\mathcal{P}(i+k,i) = \frac{1}{6} \quad \forall k = 1,2,...,6$$
- $\forall i \notin$ the first component of $Ladder \bigcup Snake$ and $94 \leq i \leq 99$
$$\mathcal{P}(100,i) = 1 - \frac{1}{6} * (100 - i - 1)$$

and any other not specified transition probability is 0.

Problem 2

Reference snakes_and_ladders_mp.py.

We do the simulation for 80000 times.



Problem 3 (Frog Puzzle)

We use the same logic in the problem of Snakes and Ladders to solve this problem. Our goal is to calculate the expected number of jumps for the frog to jump from one bank of river to the other bank of river. Hence, we first model this problem as a Markov Process which is stated as follows:

(1) Let $\mathcal{S} = \{state : 0 \leq state \leq 10\}$ be the state space and $\mathcal{T} = \{10\}$ be the set of terminal states.

(2) Denote S_t as the state for time step t.

(3) Denote $\mathcal{P} : \mathcal{N} \times \mathcal{S} \to [0,1]$ as the Transition Probability Function (source $s \to$ destination s') where

$$\mathcal{P}(s, s') = \mathbb{P}[S_{t+1} = s' \mid S_t = s] \text{ for all } t = 0, 1, 2, \ldots$$

(4) Termination: If $S_T \in \mathcal{T}$, the process terminates at time step T.

We then explicitly write the transition probability as follows:

$\forall\ 0 \leq i \leq 9$

$$\mathcal{P}(j, i) = \frac{1}{10 - i}\ \forall i < j \leq 10$$

and any other not specified transition probability is 0. We then extend the finite Markov Process to Finite Markov Reward Process to calculate the expected number of jumps. Let $R_t \in \mathcal{D}$ be a time-indexed sequence of Reward random variables for time steps $t = 1, 2, \ldots$ ($\mathcal{D} = \{-1\}$ in our problem). Clearly, all $\mathcal{P}(s, s') > 0$ should be associated with reward $R_{t+1} = -1$.

We do the simulation for 30000 times and found the expected frog jumps is 2.8424.

Reference frog_puzzle_mrp.py.

Problem 4

For this game, we want to minimize the steps we need to take to finish the game. Hence, we can set the Rewards model related to steps taken. Let $R_t \in \mathcal{D}$ be a time-indexed sequence of Reward random variables for time steps $t = 1, 2, \ldots$ ($\mathcal{D} = \{0, -1\}$ in our problem). The Transition Probability Function $\mathcal{P}_R : \mathcal{N} \times \mathcal{D} \times \mathcal{S} \to [0,1]$ is specified as follows:

$$\mathcal{P}_R(s, r, s') = \mathbb{P}[(R_{t+1} = r, S_{t+1} = s') \mid S_t = s]\, \forall t = 0, 1, 2, \ldots$$

.
Clearly, for problem 1 with 100 grids, the transition associated with ladders and snakes should be with $R_{t+1} = 0$ and other transitions (transition probability > 0) should be with reward $R_{t+1} = -1$
We do the simulation for 80000 times and found the expected number of dice rolls is 23.05.
Reference snakes_and_ladders_mrp.py.