Assignment 2:

1.) a.) 101 fields - top fields of snakes and bottom fields of ladder (8 ladders, 7 snakes, so 96 states), so state space S is:

$$S = \{i \in \mathbb{N}_0 : i \leq 100\} \setminus \{1, 4, 8, 28, 21, 50, 71, 88, 97, 95, 63, 48, 36, 32\}$$

b.) For every state i, there will be six possible outcome states depending on the roll of the dice. Each of these will be assigned a probability of $\frac{1}{6}$. The six possible outcomes will usually be the six states labeled with the numbers succeeding the label number of the current state. There are two main exceptions to this. First, one of the six fields after that represented by the current state can be a field at the bottom of a ladder or at the top of a snake. In that case $\frac{1}{6}$ of the probability is assigned to the state representing the other end of that snake or ladder. This is modelled below using L, whilst assuming all snakes and ladders span at least 6 fields. Secondly, the current state can be less than 6 fields away from 100. In this case, the probability of ending up in the terminal state 100 is above $\frac{1}{6}$, as described below.

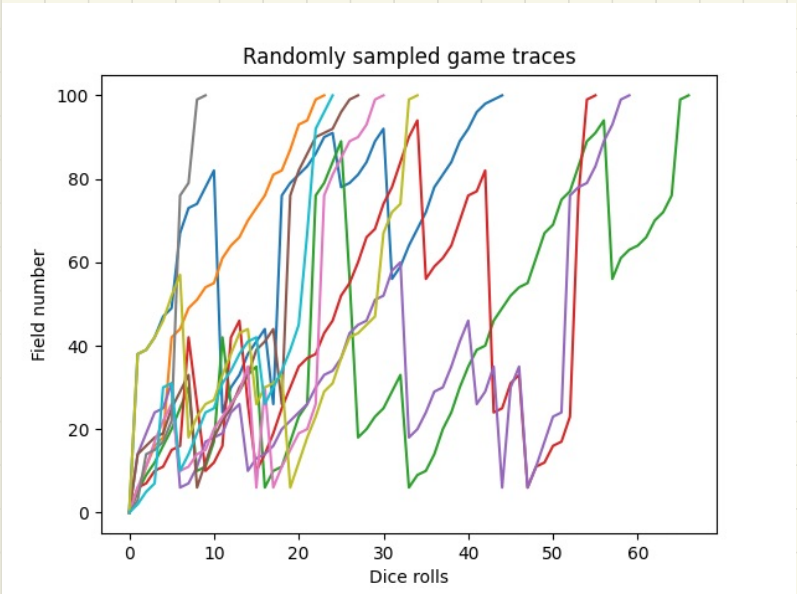
$$L = \{(1, 38), (4, 14), (8, 30), (21, 42), (28, 76), (71, 92), (80, 99), (32, 10), (36, 6), (48, 26), (63, 18), (88, 24), (95, 56), (97, 78)\}$$

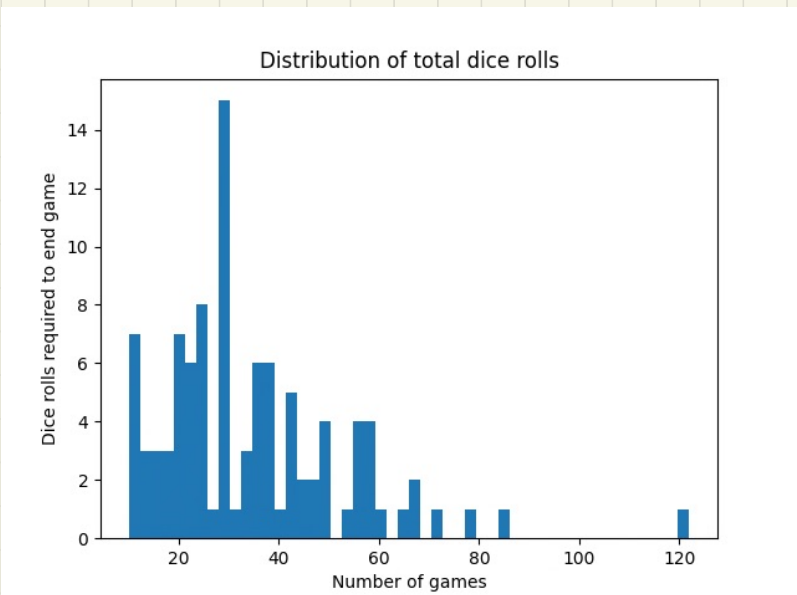
Transition probabilities:

$$\forall i, j \in S: P(S_{t+1} = j \mid S_t = i) = \begin{cases} \frac{1}{6} & \text{if } 1 \leq j - i \leq 6 \wedge j \neq 100 \\ \frac{1}{6} & \text{if } 1 \leq k - i \leq 6 \wedge (k, j) \in L \\ \frac{n}{6} & \text{if } j = 100 \wedge 7 - (j - i) = n \\ 0 & \text{else} \end{cases}$$

c.) See assignment 1.1.py

d.)





e.) See code in assignment1.1.py

Expected number of rolls is 34.0259

2.a.)

$S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ where 0 corresponds to start and 10 corresponds to end.

$T = \{10\}$

$$P(S_{t+1} = n \mid S_t = i) = \begin{cases} \frac{1}{10-i} & \text{if } i < n \leq 10 \\ 0 & \text{else} \end{cases}$$

b.) Using FiniteMarkovProcess class we get $E[X] = 2.928968\ldots$

See code in assignment1.2.py

c.) Let n be the number of lilypads, X is number of jumps

$$E[X \mid n=0] = 1$$

$$E[X \mid n=1] = \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 1 = 1.5 = 1 + \frac{1}{2}$$

$$E[X \mid n=2] = \frac{1}{3} \cdot (1+0) + \frac{1}{3}(1+1) + \frac{1}{3}(1+1.5)$$

$$= \frac{1}{3} + \frac{2}{3} + \frac{1}{3} \cdot \frac{5}{2} = \frac{6}{6} + \frac{5}{6} = 1.5 + \frac{1}{3}$$

Continuing this pattern: $E[X \mid n] = \sum_{i=1}^{n+1} \frac{1}{i}$

3. $S = \{1, 2, 3, \dots\}$ $s_0 = 1$ $a \in [0,1]$

a.) $P[s+1 | s, a] = a$ $\quad R(s+1 | s, a) = 1 - a$

$P[s | s, a] = 1 - a \quad \forall s \in S \wedge a \in A \quad R(s | s, a) = 1 + a$

$\gamma = 0.5$

$$V^*(s) = \max_{a \in [0,1]} \left\{ R(s,a) + 0.5 \sum_{s' \in \{s, s+1\}} P(s, a, s') \cdot V^*(s') \right\}$$

MRP is time-homogeneous

$$V^*(s) = \max_{a \in [0,1]} \left\{ a \cdot (1-a) + (1-a) \cdot (1+a) + 0.5 \cdot \left(a \cdot V^*(s) + (1-a) \cdot V^*(s) \right) \right\}$$

$$= \max_{a \in [0,1]} \left\{ a \cdot (1-a) + (1-a) \cdot (1+a) + 0.5 \cdot V^*(s) \right\}$$

$V^*(s') = V^*(s)$ in this process

$$= \max_{a \in [0,1]} \left\{ a - a^2 + 1 - a^2 + 0.5 V^*(s) \right\}$$

$$\frac{\partial V^*(s)}{\partial a} = 1 - 4a + 0.5 \cdot \frac{\partial V^*(s)}{\partial a}$$

$$= 2 - 8a \Rightarrow a = 0.25$$

We know this is a maximum since $\frac{\partial^2 V^*(s)}{\partial^2 a} = -8$

$$V^*(s) = 0.25 - 2 \cdot 0.25^2 + 1 + 0.5 V^*(s)$$

$$\frac{1}{2} V^*(s) = 0.125 + 1$$

$$V^*(s) = 2.25$$

b.) $\pi^*(s) = 0.25 \quad \forall s \in S$

(4.) a.)

Strategy A: $P_A(S_{t+1} = i-1 \mid S_t = i) = \frac{i}{n}$

$P_A(S_{t+1} = i+1 \mid S_t = i) = \frac{n-i}{n}$

Strategy B: $P_B(S_{t+1} = j \mid S_t = i) = \frac{1}{n} \quad \forall j: 0 \leq j \leq n$

We can model this problem as an MPP:

$S = \{0, 1, 2, 3, \dots, n\}$ $A = \{1, 2\}$ where 1 corresponds to strategy A and 2 to B.

$R(S) = \begin{cases} 1 & \text{if } S = n \\ 0 & \text{if } 0 \leq S \leq n-1 \end{cases}$ $T = \{0, n\}$

$$P(s', a, s) = \begin{cases} \frac{i}{n} & \text{if } s' = s-1 \wedge a = 1 \wedge 1 \leq s \leq n-1 \\ \frac{n-i}{n} & \text{if } s' = s+1 \wedge a = 1 \wedge 1 \leq s \leq n-1 \\ \frac{1}{n} & \text{if } a = 2 \wedge 0 \leq s' \leq n \\ 0 & \text{else} \end{cases}$$

b.) See code in assignment 1.4.py

c.) For all variants of n, the optimal policy appears to be to take strategy A for all states except for in state 1. The escape probabilities resulting from application of this policy are plotted below