UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

Chair of Algorithms for Intelligent systems
Prof. Dr.Dirk Sudholt

Thesis:

How long is a game of snakes and ladders?

05.08.2021

Edwin Ismail
Schärdinger str 56
94032,Passau
ismail04@ads.uni-passau.de

Examiner: Prof. Dr. Dirk Sudholt
Second examiner: PD Dr. Lorenz Gilch

Contents

List of Figures

List of Tables

Listings

Abstract

The game of snakes and ladders is a very famous game from ancient India. The game spiked the interest of mathematicians to construct a Markov chain model to find the expected average time to finish the game. However, most of the researches cover one parameter, which was the size of the die. This thesis aims to find the average length of the game if the distribution of snakes and ladders is random. In addition, we are using a harmonic die to find the expected average length of the game.

To test the hypothesis that harmonic die gives the shortest expected average length of the game and effects of snakes and ladders on the expected average length of the game. A pseudo-random number generator was used to create a set of random snakes and ladders. Also, Markov chain models were used to analyze the game to find the average expected length of the game. For objective analysis, the Monte-Carlo method was used to simulate the game.

The results suggested that the harmonic die gives the shortest average length of the game. Furthermore, snakes have more effects on the average length of the game than ladders.

1 Introduction

Game of snakes and ladders is a British version of an ancient Indian board game called **Moksha Patam** (figure 1). The ladders on this game symbolize honesty, which means if we do virtuous deeds, we can succeed in life, and snakes represent vice such as theft, which, when committed, descends our life to lower classes. Initially, the game had ten snakes and nine ladders; this symbolized that virtues are fewer than vice in life. The game was brought to Europe during the British occupation of India. However, the British changed the game a bit the put an equal number of snakes and ladders, representing that each vice has equal redemption.

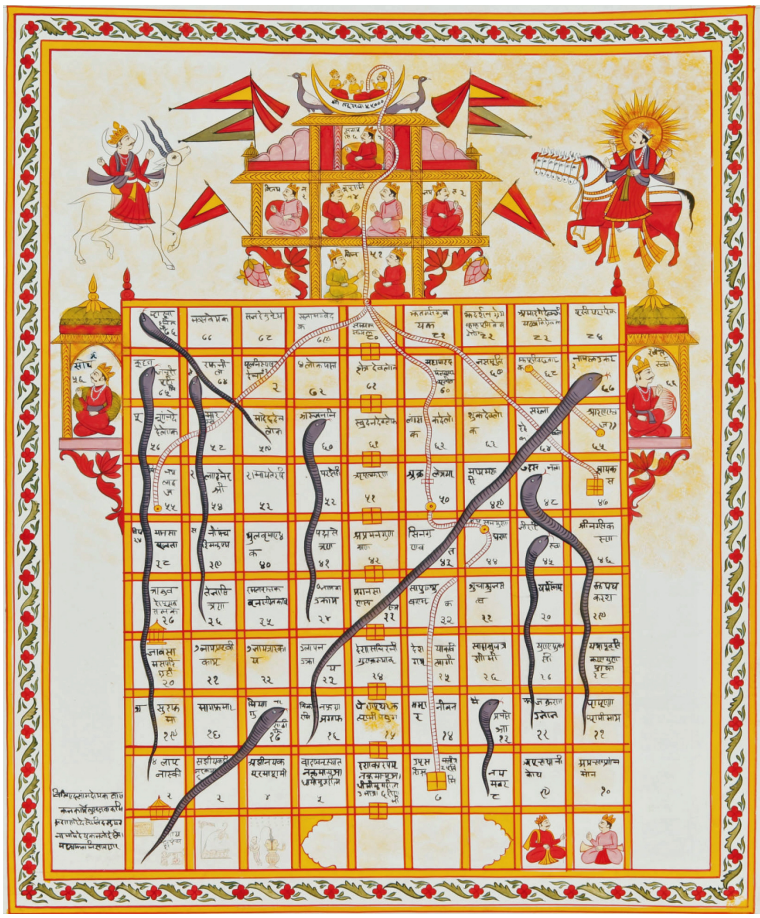


Figure 1: Moksha Patam: Original game of snakes and ladders

In USA the game was renamed as *chutes and ladders* (Figure 2) and become children favorite game. This game consists of a board with 100 square, a token representing a player, and a 6-sides die or spinner, which dictate jumps of the token. Furthermore, the board has 10 snakes and 9 ladders. The game ends when a token lands exactly at 100 square. If the token overpass 100 square, it remains in the same square before the jump. Also, if the player falls on the

bottom of the ladder, the token ascends to the top of the ladder; and when it lands on the snake's head, it will descend to the snake's tail.

Figure 2: Modern game of snakes and ladders

This game prompted a question to the mathematicians on how long does it take to finish the game? With linear algebra and probabilities, mathematicians made a Markov model and calculated the average duration to finish the game. However, scientists are still looking to make the average game length as short as possible.

The motivation for this thesis is to try to answer some of the open questions of the game. One of those questions is the average time to finish the game if the game's parameters are changed. We will study the effect of different parameters of the game on the average length of the game. The parameters are the number of snakes and ladders and the different distribution of the die.

2 Literature review

2.1 History of probability and games of chance

In ancient times, people believed that the Gods decided the occurrence of any event, and there were no scientific means to predict future events. Hence, the astrologists who could predict lunar eclipse were seen as mediums to the Gods. In gambling, winning in the game of chance was considered a favour from the Gods. However, in the seventeenth century, a French gambler named Chevalier de Mere, who was an expert in the game of chance, observed that in any four rolls of a die, six appeared at least once [GS06]. De Mère suggested this problem of winning the games of chance to the French Mathematician Blaise Pascal who communicated with another French Mathematician Pierre Fermat [DB15]. Together tried to solve the problem of games of chance, such as objects' arrangements to win the game, they came up with terms such as Mean(expected value). Pascal elaborate that if a fair coin(Head and tail) is tossed twice, the chance for the outcome would be *TT, HH, HT, TH*, he linked the possible combinatorial outcome coefficients ${}^nC_r$ with binomial coefficients $\binom{n}{r}$

$$(H+T)^2 = HH + HT + TH + TT;\ if\ (HT = TH) then \tag{1}$$

then,

$$(H+T)^2 = HH + 2HT + TT \tag{2}$$

This means, if you toss two coins the chance to get *HH or TT* is 0.25 (1/4) and getting *HT* is 0.5 . Then when tossing three coins the possible outcome were *HHH, HHT, HTH, THH, HTT, THT, TTH and TTT* and linked it to the binomial expansion of :

$$(H+T)^3 = H^3 + 3H^2T + 3HT^2 + T^3 \tag{3}$$

The coefficient derived from binomial expansion solves the possible chance for one player to win a game. For example, for the player in (equation 3) to win 3H and T is $\frac{3}{8}$. Pascal and Fermat laid the foundation of classical probability theory through their analysis, especially in objects' arrangements, i.e. combination and permutation. However, Dutch mathematician Christian Huygens through his work *De Rationiis in Ludo Aleae (On Reasoning in Games of Dice)*[DB15] defined probability of an event as the quotient of a number of occurrence of that event over all possible events.

Jacob Bernoulli also contributed to the probability theory when he introduced the ***Law of large numbers*** which is used in this thesis to find the expected length of the game. Bernoulli observed that if the experiment is done many times, the observed probability of an event will approach the theoretical probability of an even $E(X) = NP$[1] or the expected event is approximate to theoretical probability times number trials (independent). Bernoulli introduced an objective way to assign a probability of an event. His observation helped answer the *problem of gambler fallacy* which assumes that since the probability of ideal coin is 0.5, then for ten consecutive tosses, the outcome should be five heads and five tails. Hence, when setting rules for the games of chance, most

[1] *N is the number of trials and P is the theoretical probability of event X*

gambling houses use the Law of large numbers to determine the chance of them winning in the long run.

An English Mathematician, Thomas Bayes, came with another approach to assigning a probability to an event. His approach was finding a probability of an event based on a number of success and failure on an observed number of independent trials. In contrast to Bernoulli, who assigned a probability of success event k in independent trials, assuming the probability of success of each trial is p. Bayer's analysis on probability lead to the introduction of *conditional probability*. His approach is called *Subjective probability*

Siméon Poisson defined probability subjectively and objectively. Poisson stated *'The probability of an event is the rationale to believe that it has happened or that it will happen. Also, the measure of the probability of an event is the ratio of the number of cases favourable for it to the total number of favourable and contrary cases, all of them equally possible or having the same chance'* [Poi19] form his work *Recherches sur la probabilité des jugements en matière criminelle et en matière civile.*

At this stage, we can confidently state that the probability assignment of an event is achieved either objectively or subjectively. Objective assignment of event's probability is achieved through experimentation, and subjective assignment can be done through formal modelling.

2.2 Probability and Board games

In classical probability theory, we are only concerned with assigning independent events to fixed probabilities. Thus the occurrence of event E_i corresponds to fixed probability P_i. Hence, for independent events to happen in consecutive order, their probability can be defined as

$$\mathbf{P}(E_1, E_2, E_3, \ldots, E_n) = P_1 \cdot P_2 \cdot P_3 \ldots P_n$$

Nevertheless, what if the occurrence of the next event depends on the current's state probability and only it. Therefore, occurrence of E_j depends on probability conditional probability P_i. So, the sequence on this new idea is:

$$\mathbf{P}(E_1, E_2, E_3, \ldots, E_n) = P_{12} \cdot P_{23} \cdot P_{34} \ldots P_{(n-1)n}$$

This idea is called ***Markov chain*** [Fel68]

2.2.1 Markov chain Theory

Markov chains are probability models,and can be explained as follows: if we have a set of states $S = \{s_1, s_2, ..., s_n\}$. The chain can start at any of the states and move to another state, moving from state s_i to state s_j depends on probability p_{ij} which does not depend on the probability of the previous state s_{i-1}. Also state s_i can remain in the same state then probability will be denoted as p_{ii}. The probability p_{ij} or p_{ii} is called transition probability. There are many types of Markov chains, but we will discuss the finite Markov chain in this thesis. There are two types of states for the finite Markov chain: recurrent or transient states and absorbing states. The recurrent states are states that can be revisited many times during the Markov process, and absorbing states are states when reached Markov process stays there.

Example

When a student is at the university, he can go to the *klostergarten*, and from *klostergarten* he can go to the supermarket with a probability of 0.3 or bus station with a probability of 0.7. If he reaches the bus station, he can either return to *klostergarten* with the probability of 0.05 or go to the supermarket with a probability of 0.15 or take the bus with a probability of 0.8. If he is at the supermarket, the chance to go to *klostergarten* is 0.2 and to go to the bus station is 0.8. However, once he takes a bus, he can never return to any of the earlier places. The student's movement from University to mentioned places can be modelled as a Markov chain.
The above chain can be illustrated below 3:
abbreviations: University = U, Klostergaten = K, Supermarket = S, Bus Station = BS and Bus = B

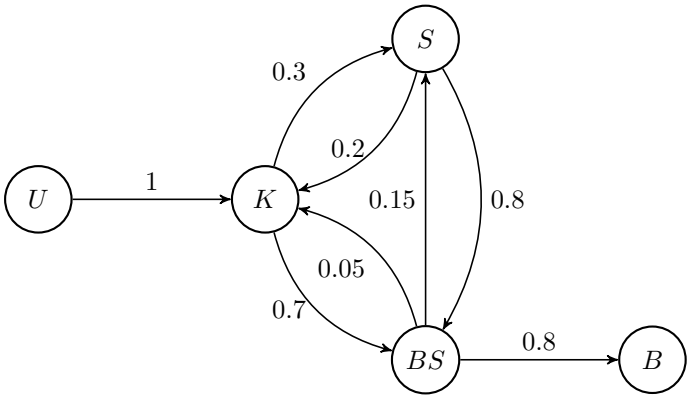


Figure 3: Markov chain for student's movement

The transition probability from the Markov chain can be presented as a square array, which can be called *transition matrix*.

$$\mathbf{P} = \begin{array}{c} \\ U \\ K \\ S \\ BS \\ B \end{array} \begin{array}{c} \begin{array}{ccccc} U & K & S & BS & B \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.3 & 0.7 & 0 \\ 0 & 0.2 & 0 & 0.8 & 0 \\ 0 & 0.05 & 0.15 & 0 & 0.8 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

State B(Bus) is an absorbing state, which indicates once reached the chain stays there infinitely. Some Interesting questions can be asked concerning the absorbing states. Such questions are: *(i)* how many times the transient state is visited before arriving at an absorbing state? *(ii)* how long will it take to reach the absorbing state on average? [GS06]. With the help of linear algebra, we can answer those questions easily.

Rearranging the transition matrix so that transient state come before absorbing state will result into following *canonical form* where there are *a* absorbing

states and t transient states

$$\mathbf{P} = \left(\begin{array}{c|c} \mathbf{Q} & \mathbf{R} \\ \hline \mathbf{0} & \mathrm{I} \end{array}\right) \tag{4}$$

Therefore, $\mathbf{Q}$ becomes $(t \times t)$ matrix, $\mathbf{R}$ is nonzero matrix of $(t \times a)$, $\mathbf{0}$ is $(a \times t)$ zero matrix and finally $\mathbf{I}$ is identity matrix of $(a \times a)$. For any absorbing Markov chain there is an inverse N of $\mathbf{I} - \mathbf{Q}$. Thus, the matrix $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$ is called ***fundamental matrix*** for transition matrix $\mathbf{P}$ whereby $\mathbf{N}$ stands for number of expected times the Markov process passes through transient state s_j, if it started from transient state s_i.

Hence, let us model student's movement in a *canonical form* so that to get the answer to the question *(i)* number of times he differently visited places before taking a bus *(ii)* how long it will take him to take a bus from any state. The canonical form of transition matrix $\mathbf{P}$ will be as follows:

$$\mathbf{P} = \begin{array}{c} \\ U \\ K \\ S \\ BS \\ \hline B \end{array} \begin{array}{c} \begin{array}{cccc} U & K & S & BS \end{array} \\ \left(\begin{array}{cccc|c} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.3 & 0.7 & 0 \\ 0 & 0.2 & 0 & 0.8 & 0 \\ 0 & 0.05 & 0.15 & 0 & 0.8 \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right) \end{array}$$

Then, we find $\mathbf{Q}$ to be

$$\mathbf{Q} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 0.7 \\ 0 & 0.2 & 0 & 0.8 \\ 0 & 0.05 & 0.15 & 0 \end{pmatrix}$$

and,

$$\mathbf{I} - \mathbf{Q} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 0.7 \\ 0 & 0.2 & 0 & 0.8 \\ 0 & 0.05 & 0.15 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -0.3 & -0.7 \\ 0 & -0.2 & 1 & -0.8 \\ 0 & -0.05 & -0.15 & 1 \end{pmatrix}$$

But

$$\mathbf{N} = (\mathbf{Q} - \mathbf{I})^{-\mathbf{1}} \tag{5}$$

then,

$$\mathbf{N} = \begin{array}{c} \\ U \\ K \\ S \\ BS \end{array} \begin{array}{c} \begin{array}{cccc} U & K & S & BS \end{array} \\ \begin{pmatrix} 1 & 1.17 & 0.539 & 1.25 \\ 0 & 1.17 & 0.539 & 1.25 \\ 0 & 0.319 & 1.28 & 1.25 \\ 0 & 0.106 & 0.219 & 1.25 \end{pmatrix} \end{array}$$

Thus, if we take look at row K in matrix $\mathbf{N}$, we can conclude that if the student is in state K(*klostergarten*) number of time he visits *university*, *klostergarten*, *supermarket*, *bus station* before he takes a bus is 0,1.17,0.54 and 1.25 respectively. We can now answer the second question: how long will it take him to

take a bus? If we add all entries to each transient state S_i, the sum will give us the expected number of stops before he takes the bus (*reaching absorbing state*). This is given by multiplying **fundamental matrix N** by vector matrix of 1 $(t \times 1)$ **v** .
So, let expected time be $\boldsymbol{T}$ then,

$$\mathbf{T} = \mathbf{Nv} \tag{6}$$

Hence,

$$\mathbf{T} = \mathbf{Nv} = \begin{pmatrix} 1 & 1.17 & 0.539 & 1.25 \\ 0 & 1.17 & 0.539 & 1.25 \\ 0 & 0.319 & 1.28 & 1.25 \\ 0 & 0.106 & 0.219 & 1.25 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3.96 \\ 2.96 \\ 2.85 \\ 1.58 \end{pmatrix}$$

From the **T** we find out that the average places the student pass on his way to take a bus is 3.96 . This answer was proven through a simulation which gives the average of 3.9637 stops after 1000000 trials. Therefore, the Markov chain analysis proves that probabilistic models can give exactly the answer as simulation (*law of large numbers*). See program 1[random walk simulation] in the appendix.

2.2.2 Board game as Markov chain

Some of the board game such as *Monopoly, Hi Ho! Cherry - O* and *Snakes and Ladders* can be analysed as a Markov chain. These games have one starting point and multiple points (squares or positions), which can be revisited during the game. Furthermore, there is a point that when reached, the game ends. Thus, we can denote those recurrent points as *transient tastes* and the point when reached the game ends as the *absorbing state*.

2.2.2.1 Hi Ho! Cherry - O

This is a board game where to win the game a player should fill his bucket with 10 cherries from the trees, this game can accommodate four players. Also, the game has a spinner with 7 equal regions, the possible results of a spin are

- **One cherry**: player picks one cherry and puts it in their bucket.
- **two cherries**: player picks two cherries and puts them in his bucket.
- **three cherries**: player picks three cherries and puts them in his bucket.
- **four cherries**: player picks four cherries and puts them in his bucket.
- **bird**: player returns two cherries from his bucket to the tree if a player has one cherry return that cherry to the tree; if he has no cherry he does nothing
- **dog**: same as bird
- **spilled bucket**: player returns all the cherries to the tree if he has none, he also returns all cherries meaning he remains in the state 0.

Figure 4: Hi Ho! cherry - O board

This game can be modelled as a Markov chain; we consider a number of cherries from 0 to 10 as states in the Markov chain whereby 0 to 9 are transient states, and 10 is an absorbing state. Moreover, the game ends when a player has ten or more cherries in his bucket. The Markov chain for this game is shown in the figure 5 below. However, we consider some of the states only for better visibility. We use linear algebra to solve some common questions like how long it takes for a player to win this game, recall subsection 2.2.1. Therefore we represent the Markov chain 5 by a transition matrix $\mathbf{P}$ in equation 7; furthermore, each row represents a state in the Markov chain.

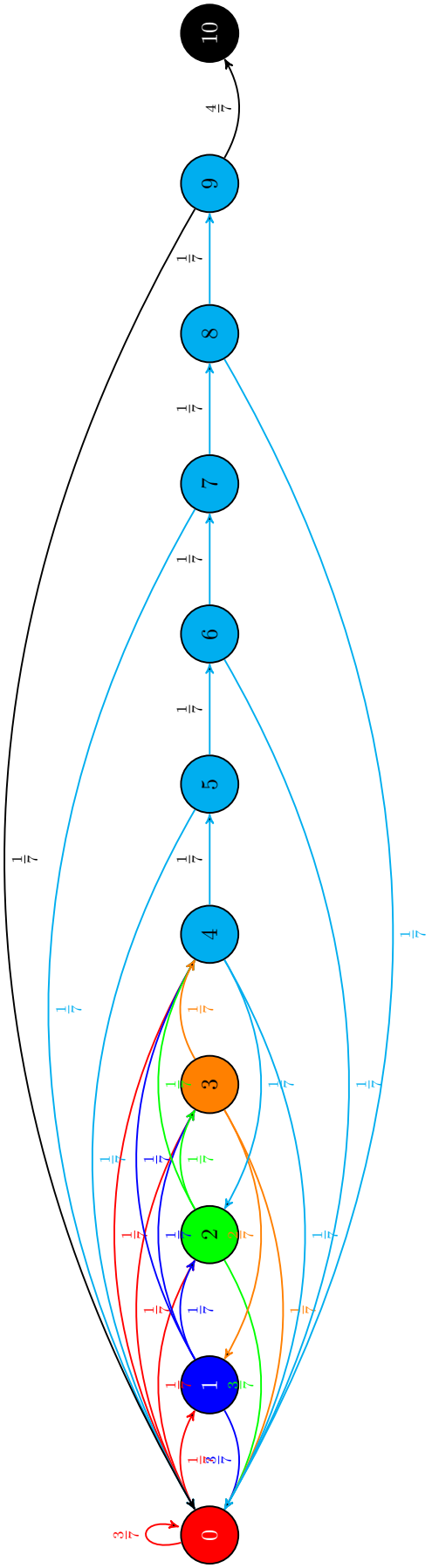


Figure 5: part of Hi Ho cherry - O Markov Chain

$$P = \begin{bmatrix} 3/7 & 1/7 & 1/7 & 1/7 & 1/7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3/7 & 0 & 1/7 & 1/7 & 1/7 & 1/7 & 0 & 0 & 0 & 0 & 0 \\ 3/7 & 0 & 0 & 1/7 & 1/7 & 1/7 & 1/7 & 0 & 0 & 0 & 0 \\ 1/7 & 2/7 & 0 & 0 & 1/7 & 1/7 & 1/7 & 1/7 & 0 & 0 & 0 \\ 1/7 & 0 & 2/7 & 0 & 0 & 1/7 & 1/7 & 1/7 & 1/7 & 0 & 0 \\ 1/7 & 0 & 0 & 2/7 & 0 & 0 & 1/7 & 1/7 & 1/7 & 1/7 & 0 \\ 1/7 & 0 & 0 & 0 & 2/7 & 0 & 0 & 1/7 & 1/7 & 1/7 & 1/7 \\ 1/7 & 0 & 0 & 0 & 0 & 2/7 & 0 & 0 & 1/7 & 1/7 & 2/7 \\ 1/7 & 0 & 0 & 0 & 0 & 0 & 2/7 & 0 & 0 & 1/7 & 3/7 \\ 1/7 & 0 & 0 & 0 & 0 & 0 & 0 & 2/7 & 0 & 0 & 4/7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Now let us discuss the above transition matrix, the first row at entry $P_{0,0}$ is $\frac{3}{7}$ since the probability for the outcome to be *spilled bucket, dog and bird* is $\frac{1}{7}$; thus if any of those events happens the bucket remains empty, i.e. stays at state 0. Also, for the second and third row, both $P_{1,0}$ and $P_{2,0}$ are $\frac{3}{7}$ because the probability of a dog or bird is $\frac{1}{7}$. Also, the spilled bucket is $\frac{1}{7}$ which will set a bucket to 0 state. To find the average time for a player to win this game can be obtained the same way as subsection 2.2.1 example nevertheless, the goal of this thesis is the game snakes and ladders. In consequence, we will not go deep on the Hi-Ho cherry - O game.

2.2.2.2 game of snakes and ladders

The game of snakes and ladders is also a board game that can be modeled as a Markov chain because it has *memory-less* property. The *momerylessness* means that the player's jumps from the current square to the next square are independent of how they reached that square. However, unlike *Hi Ho Cherry - O* to win the game, a player must fall at exactly the last square. Each square is treated as a state, and moving from one state to another is dictated by a die or spinner. Therefore, the probability of the side of the die serves as a transition probability from one state to another.

2.3 Related works

The paper [AKS93] was among the earliest papers to treat the game of snakes and ladders as Markov chain. They had done a simulation of 10 runs of 1000 games each, and the average result obtained was 39.1. However, they did a Markov chain analysis on the game and found the expected average length was 39.2. [AKS93] Markov chain analysis is the essential foundation of this thesis. We are going to see how to analyses the game of snakes and ladders as a Markov chain.

2.3.1 Length of standard snakes and ladders game

We will model snakes and ladders as Markov chain, whereby each player starts at state 0 and finishes at 100. Since the presence of more than one player does not affect the length of the game, hence we are going to model it as one player game.

Therefore, we generate a 101×101 transition matrix $\mathbf{P}$. We denote i to stand for a Markov chain state representing a square in the game, thus $0 \leqslant i \leqslant 100$; we first consider transition a matrix $\mathbf{P}$ without snakes and ladders, Then we eventually add snakes and ladders, so the new transition Matrix is $\mathbf{P}^*$.
Keep in mind that entry *(i,j)* in $\mathbf{P}$ represents transition probability from state i to state j. For the standard rule of the game, it ends when the player reaches 100; thus, square 100 is absorbing state, making entry $\mathbf{P}_{100,100} = 1$. Also, recall that the game starts at zero, which is off-board, making it impossible to visit during the game; therefore, $\mathbf{P}_{0,0} = 0$ making entire states being 101. If we do not consider the presence of the ladders and snakes, the transition matrix P will be as follows:
let n be the size of the dice or spinner

$$\text{die: the size of the die e.g 6} \tag{8}$$

$$\text{board: the size of the board of the game e.g 100} \tag{9}$$

$$\text{let } n = \text{die; } state = \text{board} + 1 \tag{10}$$

$$\text{for } 0 \leqslant i \leqslant (state - n);\ P_{i,i+j} = \frac{1}{n};\ \text{whereby } 1 \leqslant j \leqslant n\ ; \tag{11}$$

$$\text{for } (state - n) \leqslant i \leqslant board;\ P_{i,i} = \frac{n - board + i}{n} \tag{12}$$

$$\text{for } (state - n) \leqslant i \leqslant board;\ P_{i,i+j} = \frac{1}{n};\ \text{where } 1 \leqslant j \leqslant (board - i) \tag{13}$$

The transition probability is $\frac{1}{n}$ for all rows from 0 to $(100-n)$ if the corresponding column is less than or equal to size of dice (n) from (11) and in equation (13) for all rows from $(101-n)$ to 100; whereby the corresponding columns are between range of $(101-n)$ to 100. Equation (12) gives transition probabilities for all diagonal entries.

When we consider the presence of snakes and ladders, the transition matrix changes, we denote it as P^*. Hence, all rows and columns which is the bottom of the ladder or the head of the snake become zero since the token never stays at those states. However, we add the transition probability which was from the top of the snake or bottom of the ladder to the transition probability at the bottom of the snake or the top of the ladder. Recall, if token fall on bottom of the ladder it moves to the top of the ladder and when falls on the top of the snake it moves down to the end of the snake.

If we assign $[i, j]$ as snake or ladder, then algorithm for transition matrix P^* from transition matrix P will be as follows:

$$\text{for } P^*_{i,k} = 0;\ \text{for all } k \tag{14}$$

$$\text{for } P^*_{k,j} = P^*_{k,j} + P^*_{k,i};\ \text{where } i \neq k \tag{15}$$

$$\text{for } P^*_{k,i} = 0;\ \text{for all } k \tag{16}$$

We use a small board for simplicity and better presentation of Markov chain analysis of the game. However, the small board is used only in this chapter. The results from other chapters are based on the analysis of the original game. Hence, our board will be 4×4, with 2 snakes ($[6, 3], [14, 7]$) and one ladder $[4, 13]$. see figure 6 below

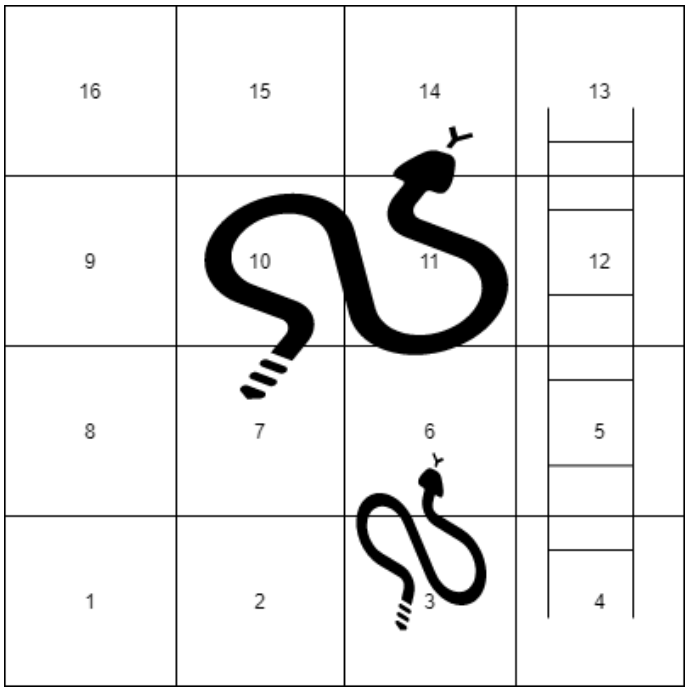


Figure 6: simple snakes and ladders board

We assume our die is of size 4, with equally distributed probability. A player starts from state 0 to 16, generating 17×17 transition matrix. We construct a transition matrix ignoring snakes and a ladder, using algorithm from eqs. (11) to (13). We notice that rows 12 and 13 are identical. The reason if the token is at position 12, token moves to $13, 14, 15, 16$. However, for position 13, the token remains at 13 with the probability of $\frac{1}{4}$ when the outcome is 4, meaning it has passed 16 square. For an outcome of $1, 2, 3$ the token moves to $14, 15, 16$ with probability $\frac{1}{4}$ each.

$$\mathbf{P} = \begin{pmatrix}
0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.5 & 0.25 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.75 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.
\end{pmatrix}$$

Now, let us consider the presence of snakes and ladders. Recall, when the token lands at the bottom of the ladder, it ascends to the top, and if the token falls on the head of the snake, it descends to its tail. Thus, a token never stays at the head of the snake or at the bottom of the ladder, making the corresponding

row and column 0. Therefore, we take the probability at the bottom of the ladder or top of the snake and add it with the probability of the square at the bottom of the snake or the top of the ladder. Thus, forming a new transition matrix P^* refer eqs. (14) to (16).

$$\mathbf{P}^* = \begin{pmatrix}
0. & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. \\
0. & 0. & 0.25 & 0.25 & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. \\
0. & 0. & 0. & 0.5 & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. \\
0. & 0. & 0. & 0.25 & 0. & 0.25 & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0. & 0.25 & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0.25 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0.25 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.75 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.
\end{pmatrix}$$

Since, the Markov chain in the game snakes and ladders is *absorbing* [AKS93; Joh03], we can rearrange transition matrix P^* into canonical form eq. (4). So, $\mathbf{Q}$ is 16×16 matrix from $P^*_{0,0}$ to $P^*_{15,15}$.

$$\mathbf{Q} = \begin{pmatrix}
0. & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. \\
0. & 0. & 0.25 & 0.25 & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. \\
0. & 0. & 0. & 0.5 & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. \\
0. & 0. & 0. & 0.25 & 0. & 0.25 & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0. & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0. & 0. & 0. & 0. & 0.25 & 0. & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.75
\end{pmatrix}$$

Recall fundamental form eq. (5), so

$$\mathbf{N} = \begin{pmatrix}
1. & 0.25 & 0.3125 & 0.733 & 0. & 0.324 & 0. & 0.930 & 0.314 & 0.392 & 0.409 & 0.511 & 0.406 & 1.338 & 0. & 2.256 \\
0. & 1. & 0.25 & 0.659 & 0. & 0.477 & 0. & 0.964 & 0.360 & 0.450 & 0.444 & 0.555 & 0.452 & 1.270 & 0. & 2.277 \\
0. & 0. & 1. & 0.818 & 0. & 0.455 & 0. & 1. & 0.364 & 0.456 & 0.456 & 0.570 & 0.461 & 1.254 & 0. & 2.285 \\
0. & 0. & 0. & 1.455 & 0. & 0.364 & 0. & 1.159 & 0.381 & 0.476 & 0.504 & 0.630 & 0.498 & 1.187 & 0. & 2.315 \\
0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0.364 & 0. & 1.09 & 0. & 1.1 & 0.548 & 0.685 & 0.583 & 0.729 & 0.636 & 0.999 & 0. & 2.365 \\
0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.783 & 0.446 & 0.557 & 0.696 & 0.870 & 0.642 & 0.922 & 0. & 2.435 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.745 & 1.186 & 0.483 & 0.604 & 0.754 & 0.7575 & 0.866 & 0. & 2.377 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.715 & 0.179 & 1.223 & 0.529 & 0.662 & 0.648 & 1.021 & 0. & 2.331 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.929 & 0.232 & 0.291 & 1.363 & 0.703 & 0.647 & 1.001 & 0. & 2.352 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.743 & 0.186 & 0.232 & 0.290 & 1.363 & 0.518 & 0.801 & 0. & 2.681 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.594 & 0.149 & 0.186 & 0.232 & 0.290 & 1.214 & 0.6419 & 0. & 2.145 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.594 & 0.149 & 0.186 & 0.232 & 0.290 & 0.214 & 1.641 & 0. & 2.145 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 4.
\end{pmatrix}$$

To get average length for the whole game is multiplying matrix $\mathbf{N}$ with 16×1

vector of 1, recall eq. (6). The result is

$$\mathbf{T} = \begin{pmatrix} 9.17531196 \\ 9.16033082 \\ 9.12188044 \\ 8.96807893 \\ 1. \\ 9.10040627 \\ 1. \\ 8.3528729 \\ 7.77248984 \\ 7.3081834 \\ 7.5171213 \\ 6.81369704 \\ 5.45095763 \\ 5.45095763 \\ 1. \\ 4. \end{pmatrix} \quad \mathbf{T_{sim}} = \begin{pmatrix} 9.175709 \\ 9.147952 \\ 9.120856 \\ 8.969347 \\ 9.100149 \\ 8.353487 \\ 7.772945 \\ 7.302218 \\ 7.514459 \\ 6.809142 \\ 5.45115 \\ 5.456408 \\ 3.997515 \end{pmatrix}$$

Therefore, the expected average time to finish the game start from 0 approximately equals 9. Each row signifies the length of the game if we start from that state. However, 1 means that the row is either head of the snake or the bottom of the ladder. If we had removed them in the transition matrix, they would never appear in the matrix $\mathbf{T}$. On $\mathbf{T}_{sim}$, we can clearly see that the simulation [1] give as close results as the Markov chain model.

Algorithm 1 snakes and ladder simulation

```
count ← 0
token ← 0
while token < board do
    roll ← random(1, dice)
    token ← token + roll
    count ← count + 1
    if token > board then
        token ← token − roll
    end
    foreach snake in snakes do
        if token ≡ snake[0] then
            token ← snake[1]
        end
    end
end
return count
```

The paper [AKS93] also mentioned that the game is sensitive to the addition of the ladders and snakes. However, adding snakes or ladders can shorten or lengthen the game. For example, adding a snake from square 29 to 27 shortens the game since there is a chance to land on square 28, leading to square 84. Nevertheless, it has not mentioned the effect of the length of the snakes or ladder.

Article [LJ11] analysed game of snakes and ladders to find the average length of the game by varying the size of the die with a range of 2 to the board's size. The Markov chain analysis showed that the die size affects the game, but the die's with a large size does not result in shortening the game. Table 1 we found out spinner with size 15 (uniformly distributed) gives the best minimum expected average length of 25.81. Also, they noticed that for the board on n size, the expected length of the game is the same for dice with size n and $n-1$. The article [CG14] proved through combinatorics and recursion that for any snakes and Ladders game board with n squares, the expected average length for a token starting at square 0 lands exactly on square n is the same if one uses a die of range n or $n-1$. The article [LJ11] left an open question: how to shorten the game's length regarding the different distributions of spinners or different arrangements of the snakes and ladders.

n	E	n	E	n	E	n	E	n	E
1	∞	21	27.53	41	40.30	61	56.81	81	73.46
2	60.76	22	27.58	42	41.05	62	57.67	82	74.40
3	65.90	23	28.12	43	41.84	63	58.53	83	75.34
4	54.49	24	28.78	44	42.74	64	59.45	84	76.29
5	45.56	25	29.10	45	43.54	65	60.33	85	77.23
6	39.23	26	29.72	46	44.35	66	61.19	86	78.16
7	34.70	27	30.07	47	45.15	67	62.07	87	79.05
8	31.85	28	30.64	48	45.98	68	62.94	88	79.99
9	30.30	29	31.46	49	46.85	69	63.77	89	80.91
10	28.77	30	32.14	50	47.67	70	64.66	90	81.85
11	27.43	31	32.90	51	48.53	71	65.59	91	82.80
12	27.02	32	33.61	52	49.38	72	66.50	92	83.74
13	26.22	33	34.32	53	50.25	73	67.39	93	84.64
14	25.98	34	35.05	54	51.03	74	68.22	94	85.57
15	25.81	35	35.77	55	51.87	75	69.12	95	86.46
16	25.84	36	36.50	56	52.55	76	69.88	96	87.43
17	25.97	37	37.26	57	53.40	77	70.76	97	88.39
18	26.39	38	38.05	58	54.24	78	71.65	98	89.28
19	26.71	39	38.83	59	55.12	79	72.56	99	90.28
20	27.21	40	39.54	60	55.98	80	72.55	100	90.28

Table 1: The expected number E of turns for one player to complete Snakes and Ladders game, as a function of spinner range n
[LJ11]

The article [Die+10] proved that the minimum expected time $\mathbf{T}$ for a discrete Markov chain to reach the absorbing state is $\theta((\log n)^2)$. n represent number of states in the Markov chain. They assume the Markov chain has states $S = \{0, \ldots, n\}$ and to move from one state to another is influenced by step size d which is randomly chosen based on harmonic distribution in the set $D =$

$\{1, \ldots, n\}$.

$$\mathbf{E}_{\mu}(T) = O((log\ n)^2);\ whereby\ n \to \infty \tag{17}$$

$$H_n = \sum_{d=1}^{n} \frac{1}{d} \tag{18}$$

$\theta((\log n)^2)$ is the complexity of an algorithm, meaning the time required by an algorithm for an input of a given size (n) in their case, it is the number of states. However, the presence of snakes and ladders might affect equation 17. Hence, we want to test their method if it gives a shorter time for the Markov analysis of the game of snakes and ladders. Furthermore, answering the open question in the paper [LJ11] which is *what is the expected length of the game when using a non-uniform die*. In this thesis, we use harmonic distribution for the spinner to find the average expected length of the game.

3 Methodology

3.1 Research question

3.1.1 Effects of snakes and ladders on the average length of the game

The article [AKS93] showed that the average length of the game is 39.2. However, there is no extensive research on the average length of the game if the distribution of snakes and ladders is changed. Therefore, in this thesis, we research the average length of the game when the game has a different distribution of the snakes and ladders. In addition, we will research the effect of the length of the snake or ladder on the average length of the game.

3.1.2 Effects on die distribution on the average length of the game

The article [LJ11] used different sizes of the dice and studied the average length of the game on each size of the die. However, the [LJ11] only used uniform distribution of the die. Thus, in this thesis, we study the effect of the size of the die but with a different type of distribution such as harmonic distribution as proven in article [Die+10] to have the shortest time for a discrete random process.

3.2 Game ends, if the token lands beyond board's size

If we deviate from standard rules for the game, *the game only ends, when the token falls precisely on the final square (100 for standard board).* Now, the number of the absorbing states will increase from 1 to $dice - 1$ or we do not account for the entries probability that exceed 100; i.e. if we consider we are at rows greater than 94 (for standard game). Also, the transition matrix changes since the token does not remain in the same square if it progresses beyond the final square. Hence, the square from $board's\ size - dice$ will land on any absorbing state with an equal probability according to the size of the dice. The

transition matrix is given through algorithm Equations (22) and (23), where we ignore the presence of snakes and ladders.

$$\text{board is the size of the for example 16} \tag{19}$$

$$\text{dice stands for the size of dice e.g 4} \tag{20}$$

$$\text{let } n = \text{dice};\ state = \text{board } +\ n \tag{21}$$

$$\text{for } 0 \leqslant i < board;\ P_{i,i+j} = \frac{1}{n};\ for\ 1 \leqslant j \leqslant\ n \tag{22}$$

$$\text{for } board \leqslant i \leqslant state;\ P_{i,i} = 1 \tag{23}$$

The transition probability is $\frac{1}{n}$ from row 0 to the row less than the size of the board and the corresponding columns can be above board size due to new absorbing states $(dice - 1)$ that is $P_{100,104}$ Equation (22). From the rows which are greater than board's size their transition probability is always 1 since they are absorbing states (Equation (23))

When we consider presence of snakes and ladders the transition matrix follows the same algorithm for original game Equations (14) to (16). For our demo example Figure 6, the transition matrix $\mathbf{P}$ become a 22×22 matrix, for this new rule. The same fundamental matrix formulae Equation (5) give the expected average length of the game in this rule.

$$\mathbf{P} = \begin{pmatrix}
0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & 0.25 & 0.25 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.
\end{pmatrix}$$

3.3 Using two dice on the game

Since we have deviated from the standard rule, let us introduce a new rule in which we use pair of dice of equal size. The important thing to keep in mind is how we want our game to end because the minimum outcome of two dice is 2, making a game impossible to finish if the token lands on square 99. Hence, we can opt for several ways to end the game

- game ends as long as token reaches final square or exceed
- making extra die start from 0, meaning 0 *to* $(n - 1)$, i.e. 0 to 5, where n is the size of a dice
- game ends at exactly final square or imaginary square, which is the addition of minimum outcome of the pair of the dice.

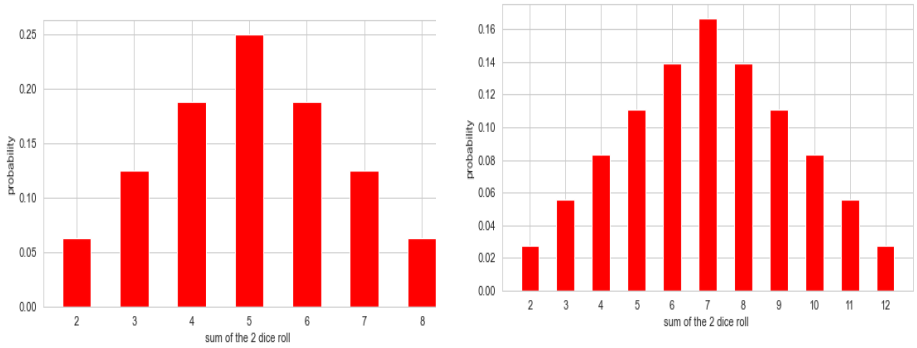
Every option mentioned produced different transition matrix $\mathbf{P}$. Also, the probability of moving the token differs from the previous. Previously, we used $\frac{1}{n}$ as

an input probability. However, right now our concern is sum of the dice. For the 2 dice with 6 sides, the distribution of the sum of the outcomes is as follows

Probability	0.028	0.056	0.083	0.111	0.139	0.167	0.139	0.111	0.083	0.056	0.0278
Sum of two dice	2	3	4	5	6	7	8	9	10	11	12

For our demo where the size of the die is 4 , the sum of probability is given below:

Probability	0.065	0.125	0.187	0.25	0.187	0.125	0.0625
Sum of two dice	2	3	4	5	6	7	8



(a) 4-sides pair of dice

(b) 6-sides pair of dice

Figure 7: probability distributions of different dice

It is tedious to find probabilities as the size of the die or number of dice increases hence, application of binomial expansion is useful in this case eq. (3). We can compute the probability of outcome t after roll of n number s-side dice as follow. The number of occurrence t is the coefficient x^t in the $f(x)$ [Usp37]

$$f(x) = (x^1 + x^2 + x^3 + \cdots + x^s)^n \tag{24}$$

we can write $f(x)$ as *multinominal series* since each arrangement contribute one term [Wei]. Hence,

$$\begin{aligned} f(x) &= x^n \left(\sum_{i=0}^{s-1} x^i \right)^n \\ &= x^n \left(\frac{1-x^s}{1-x} \right)^n, \end{aligned} \tag{25}$$

so the desired number c is the coefficient of x^t in

$$x^n \left(1-x^s\right)^n (1-x)^{-n}. \tag{26}$$

Expanding,

$$x^n \sum_{k=0}^{n} (-1)^k \left(\begin{array}{c} n \\ k \end{array} \right) x^{sk} \sum_{l=0}^{\infty} \left(\begin{array}{c} n+l-1 \\ l \end{array} \right) x^l, \tag{27}$$

so, in order to get the coefficient of x^t, include all terms with

$$t = n + sk + l. \tag{28}$$

c is therefore

$$c = \sum_{k=0}^{n} (-1)^k \left(\begin{array}{c} n \\ k \end{array} \right) \left(\begin{array}{c} t-sk-1 \\ t-sk-n \end{array} \right) . \tag{29}$$

But $t - sk - n > 0$ only when $k < (t-n)/s$, so the other terms do not contribute. Furthermore,

$$\left(\begin{array}{c} t-sk-1 \\ t-sk-n \end{array} \right) = \left(\begin{array}{c} t-sk-1 \\ n-1 \end{array} \right) \tag{30}$$

So

$$c = \sum_{k=0}^{\lfloor (t-n)/s \rfloor} (-1)^k \left(\begin{array}{c} n \\ k \end{array} \right) \left(\begin{array}{c} t-sk-1 \\ n-1 \end{array} \right) \tag{31}$$

where $\lfloor x \rfloor$ is the floor function, and

$$P(t,n,s) = \frac{1}{s^n} \sum_{k=0}^{\lfloor (t-n)/s \rfloor} (-1)^k \left(\begin{array}{c} n \\ k \end{array} \right) \left(\begin{array}{c} t-sk-1 \\ n-1 \end{array} \right) \tag{32}$$

for our demo we consider pair of dice of size 4. Hence, $s = 4$, $n = 2$

$$k_{\max} \equiv \left\lfloor \frac{t-2}{4} \right\rfloor = \left\{ \begin{array}{ll} 0 & \text{for } 2 \leqslant t \leqslant 5 \\ 1 & \text{for } 6 \leqslant t \leqslant 8, \end{array} \right. \tag{33}$$

so,

$$\begin{aligned} P(t,2,4) &= \frac{1}{4^2} \sum_{k=0}^{k_{\max}} (-1)^k \left(\begin{array}{c} 2 \\ k \end{array} \right) \left(\begin{array}{c} p-4k-1 \\ 1 \end{array} \right) \\ &= \frac{1}{4^2} \sum_{k=0}^{k_{\max}} (-1)^k \frac{2!}{k!(2-k)!} (t-4k-1) \\ &= \frac{1}{16} \sum_{k=0}^{k_{max}} (1-2k)(k+1)(t-4k-1) \\ &= \frac{1}{16} \left\{ \begin{array}{ll} t-1 & \text{for } 2 \leqslant t \leqslant 5 \\ 9-t & \text{for } 6 \leqslant t \leqslant 8 \end{array} \right. \\ &= \frac{4-|t-5|}{16} \text{ for } 2 \leqslant t \leqslant 8. \end{aligned} \tag{34}$$

for the case of dice with size of 6 each the probability will be

$$P(t,2,6) = \frac{1}{6^2}\sum_{k=0}^{k_{\max}}(-1)^k \begin{pmatrix} 2 \\ k \end{pmatrix}\begin{pmatrix} p-6k-1 \\ 1 \end{pmatrix}$$
$$= \frac{6-|t-7|}{36} \text{ for } 2 \leqslant t \leqslant 12.$$

3.3.1 Making extra die start from 0

Since we now know how to compute probabilities of inputs,Equations (25) to (34). Recall, one die starts from 0 to 5. Hence, we can achieve the original rule that's game ends when token lands on final square exactly. Thus, transition matrix $\mathbf{P}$ is:

$$\mathbf{P} = \begin{pmatrix}
0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.1875 & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.375 & 0.0625 & 0.125 & 0.1875 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.625 & 0.0625 & 0.125 & 0.1875 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.8125 & 0.0625 & 0.125 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.9375 & 0.0625 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.
\end{pmatrix}$$

The algorithm for constructing the transition matrix for this case is as follow:

let $n = dice$; $state = \text{board} + (2 \times n)$

for $0 \leqslant i \leqslant board$; $P_{(i,i+j-1)} = \dfrac{n \ - \ |j \ - \ (n+1)|}{n^2}$; whereby $2 \leqslant j \leqslant (2 \times n)$;

for $0 \leqslant i \leqslant board$; $P_{i,i} = 1 - \displaystyle\sum_{j=0}^{n=board} P_{i,j}$

The *pseudo-code* for above algorithm is as follows:

Algorithm 2 Pseudo code for transition matrix on case 3.3.1

$dice = 2 \times dice$
$state = board + 1$
P=$state \times state$ matrix
$sum = 0$
for $i \leftarrow 0$ **to** $state$ **do**
  for $j \leftarrow 2$ **to** $state$ **do**
    if $j \leqslant dice$ **and** $i \leqslant (board)$ **and** $(i + j) \leqslant P's\ size$ **then**
      $P_{i,(i+j-1)} \leftarrow \frac{dice\ -\ |j\ -\ (dice+1)|}{dice^2}$
    end
  end
end
for $i \leftarrow 0$ **to** $state$ **do**
  for $j \leftarrow 1$ **to** $state$ **do**
    $sum \leftarrow sum + P_{i,j}$
  end
  $P_{i,i} \leftarrow 1\ -\ sum$
end
return P

3.3.2 Game ends when token reached final square or exceed

As we have changed the standard rule for the game to accommodate the extra die we can set the square 16 and imaginary square 17 as the absorbing state. We include the square imaginary 17 because if the token lands on square 15 then the game will never end since the minimum step is 2 which is beyond final square hence token remains at square 15. On generating the transition matrix, we consider that state S_1 is never visited since the minimum sum is 2 hence making the whole row 0. Hence, the transition matrix P for this case is given by the following algorithm:

let $n\ = dice; state =$ board $+ 2$;

for $0 \leqslant i < (state - (2 \times n))$; $P_{i,i+j} = \dfrac{n\ -\ |j\ -\ (n+1)|}{n^2}$; whereby $2 \leqslant j \leqslant (2 \times n)$;

for $(state - (2 \times n)) \leqslant i < state$; $P_{i,i+j} = \dfrac{n\ -\ |j\ -\ (n+1)|}{n^2}$; whereby $2 \leqslant j$ **and** $i + j < state$

for $(state - (2 \times n)) \leqslant i < state$; $P_{i,i} = 1 - \sum_{j=0}^{n=board} P_{i,j}$

From above algorithm generate the below transition matrix **P**:

$$\mathbf{P} = \begin{pmatrix}
0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0625 & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.1875 & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.375 & 0. & 0.0625 & 0.125 & 0.1875 & 0.25 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.625 & 0. & 0.0625 & 0.125 & 0.1875 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.8125 & 0. & 0.0625 & 0.125 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.9375 & 0. & 0.0625 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.
\end{pmatrix}$$

3.4 Randomizing snakes and ladders

In this thesis, we also study the effects of a number of snakes and ladders on the expected average length of the game. We first study the expected length by randomizing snakes and ladders with different length. Also, we study their effects if we maintain constant length. However, the most important thing when it comes to randomization is choosing a random number generator.

All simulations in this thesis are python based program, and the most common random number generator function in python is *random*. *Random* is a pseudo-random number generator that implements the *Mersenne Twister* as a core generator whereby it produces random float number in semi-open range [0.0,1.0).

For choosing a random snake or ladder, we consider that once a number is chosen will never be chosen again, to prevent an endless loop because if the head of the snake and bottom of the ladder are at the same square making the unknown choice whether token descends to the bottom of the snake or move on top of the ladder. Also, another reason for picking a number from a pool of 1 *to* 100 without replacement is to prevent two snakes from having their heads at the same square and their tail to different square. Furthermore, We ensure that no snake's head is at the final square of the board to prevent endless game; algorithm 4.

Algorithm 3 Pseudo code for randomizing snakes and ladders.

$numberSnake, numberLadder \leftarrow input$
$randSL \leftarrow \{\}$
$countSnake, countLadder \leftarrow 0$
while $n(randSL) < (numberSnake + numberLadder)$ **do**
  $x \leftarrow randomInt(1, 100)$
  $y \leftarrow randomInt(1, 100)$
  if $y > x$ **and** $countsnake < numberSnake$ **and** $y \neq 100$ **and** $\{y, x\} \notin randSL$ **then**
    $randSL \leftarrow randSL + \{y, x\}$
    $countSnake \leftarrow countSnake + 1$
  end
  if $y < x$ **and** $countLadder < numberLadder$ **and** $\{y, x\} \notin randSL$ **then**
    $randSL \leftarrow randSL + \{y, x\}$
    $countLadder \leftarrow countLadder + 1$
  end
end

Algorithm 4 Pseudo code for randomizing snakes and ladders for fixed length.

when specify the length of the snake or ladder
$length \leftarrow input$
$numberSnake, numberLadder \leftarrow input$
$randSL \leftarrow \{\}$
$countSnake, countLadder \leftarrow 0$
while $n(randSL) < (numberSnake + numberLadder)$ **do**
  $x \leftarrow randomInt(1, 100)$
  $y \leftarrow randomInt(1, 100)$
  if $y > x$ **and** $(y - x) \equiv length$ **and** $countsnake < numberSnake$ **and** $y \neq 100$ **and** $\{y, x\} \notin randSL$ **then**
    $randSL \leftarrow randSL + \{y, x\}$
    $countSnake \leftarrow countSnake + 1$
  end
  if $y < x$ **and** $(x - y) \equiv length$ **and** $countLadder < numberLadder$ **and** $\{y, x\} \notin randSL$ **then**
    $randSL \leftarrow randSL + \{y, x\}$
    $countLadder \leftarrow countLadder + 1$
  end
end

3.5 Length of the game, if die's distribution is harmonic

The article [Die+10] suggested that choice of distribution for the set of step size is a strategy to achieve shortest average expected time to finish discrete process. Since, our game is a discrete Markov chain, therefore we are going to use the harmonic distribution on the die to find the average expected length of the game. The harmonic distribution for the die is given as follows:

- choose the size of the die n
- then find the harmonic number of that size of the die $\mathbf{H_n}$ eq. (18)

- Harmonic distribution for the elements(d) from 1 to n is :

$$\mu_{harm}(d) = \frac{1}{d \times H_n} \; for \; 1 \leqslant d \leqslant n \tag{35}$$

So the harmonic distribution for a demo with the die of size 4 is:

$$\mu_{harm_4} = [0.48, 0.24, 0.16, 0.12]$$

However, [Die+10] emphasize that to achieve the shortest average time, the range of the step size should be the same as the board. Hence, we set 16 as the side of the die. For harmonic distribution, the small steps has high probability compared to the significant steps, hence theoretically small steps are like to be chosen much more in contrast to uniform distribution resulting in reaching absorbing state earlier if a token is few steps away from reaching final state. For example, in the figure 6 if the token is at square 15, the probability of remaining at the same square is 0.9375 if we choose 16 as the size of a fair die compared to 0.7042 for a die with the harmonic distribution.

$$\mu_{harm}(die) = [0.296, 0.148, 0.099, 0.074, 0.059, 0.049, 0.042, 0.037, 0.033, 0.030, 0.027, 0.025, 0.023, 0.021, 0.020, 0.018]$$

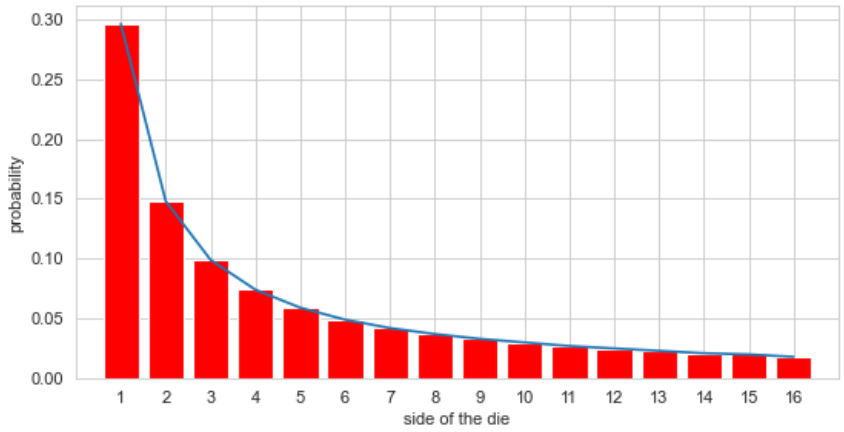


Figure 8: Harmonic distribution for the die of size 16

therefore the transition matrix uses the following algorithm:

$$\text{let } n = dice; \; board \leftarrow board's \; size; \; \mu_{harm_n} \tag{36}$$

$$\text{for } 0 \leqslant i \leqslant (board - n); \; P_{(i,i+j)} = \mu_{harm_n}(j); \; \text{whereby } 1 \leqslant j \leqslant n; \tag{37}$$

$$\text{for } (board - n) < i \leqslant board; \; P_{(i,i+j)} = \mu_{harm_n}(j); \; \text{whereby } 1 \leqslant j \leqslant (board - i) \tag{38}$$

$$\text{for } 0 \leqslant i \leqslant board; \; P_{i,i} = 1 - \sum_{j=1}^{n=board} P_{i,j} \tag{39}$$

the pseudo-code for above algorithm is :

Algorithm 5 pseudo-code for transition matrix with harmonic distribution

$dice = dice$
$board = board's\ size$
$dist = \mu_{harm}(dice)$
for $0 \leqslant i \leqslant board$ **do**
  for $1 \leqslant j \leqslant board$ **do**
    if $j \leqslant dice$ ***and*** $i \leqslant (board - dice)$ **then**
      $\mathbf{P}_{i,j} = dist[j]$
    end
    if $i > board - dice$ **then**
      if $j + i \leqslant board$ **then**
        $\mathbf{P}_{i,j} = dist[j]$
      end
    end
  end
end
for $0 \leqslant i \leqslant board$ **do**
  for $1 \leqslant j \leqslant board$ **do**
    $\mathbf{P}_{i,i} = 1 - \sum(P_{i,j})$
  end
end

The transition matrix with harmonic distribution is given below,however the sum on each row might not sum up to 1 as is supposed to be due to approximation:

$$\mathbf{P} = \begin{pmatrix}
0. & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 & 0.037 & 0.033 & 0.030 & 0.027 & 0.025 & 0.023 & 0.021 & 0.020 & 0.018 \\
0. & 0.018 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 & 0.037 & 0.033 & 0.030 & 0.027 & 0.025 & 0.023 & 0.021 & 0.020 \\
0. & 0. & 0.038 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 & 0.037 & 0.033 & 0.030 & 0.027 & 0.025 & 0.023 & 0.021 \\
0. & 0. & 0. & 0.059 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 & 0.037 & 0.033 & 0.030 & 0.027 & 0.025 & 0.023 \\
0. & 0. & 0. & 0. & 0.082 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 & 0.037 & 0.033 & 0.030 & 0.027 & 0.025 \\
0. & 0. & 0. & 0. & 0. & 0.107 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 & 0.037 & 0.033 & 0.030 & 0.027 \\
0. & 0. & 0. & 0. & 0. & 0. & 0.134 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 & 0.037 & 0.033 & 0.030 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.163 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 & 0.037 & 0.033 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.196 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 & 0.037 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.233 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 & 0.042 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.275 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 & 0.049 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.325 & 0.296 & 0.148 & 0.099 & 0.074 & 0.059 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.384 & 0.296 & 0.148 & 0.099 & 0.074 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.458 & 0.296 & 0.148 & 0.099 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.556 & 0.296 & 0.148 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.704 & 0.296 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.
\end{pmatrix}$$

3.6 Result analysis

In this thesis, all results are generated by Markov chain analysis. We use Markov chain analysis because it is fast. Since we have many sets of ladders and snakes, if we conduct a simulation for each set of snakes and ladders. The simulation takes a long time because each set of snakes and ladders is simulated 100000 times. In addition to the fast speed of Markov chain analysis, it also gives results equal to simulation (see Figure 9).

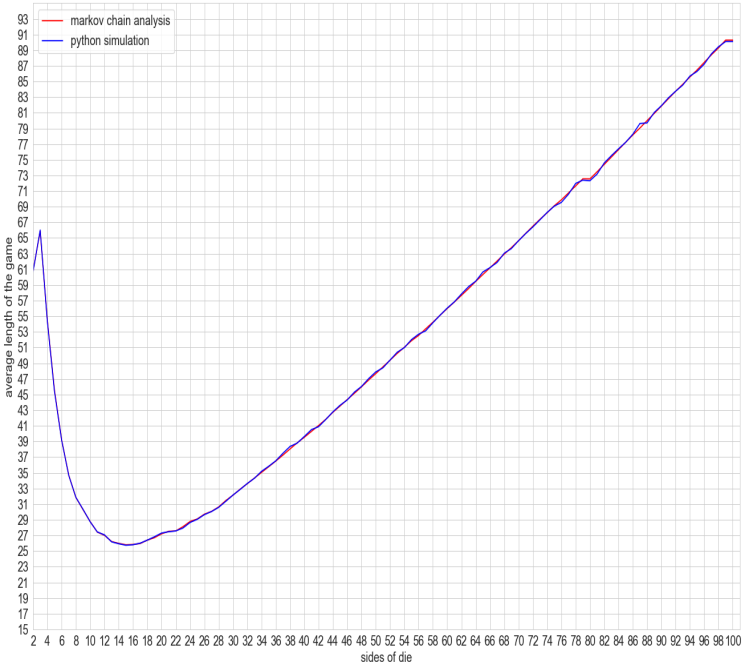


Figure 9: Average game length for simulation and Markov chain analysis using uniform die

4 Results

4.1 Effect of number of snakes and ladders

To study effect of snakes and ladders in the game, we generated 1000 sets of random ladders and snakes; meaning for one ladder and snake we had 1000 different combinations. Then, we computed the mean of the expected average length of each set of snakes and ladders.

4.1.1 Influence of number of snakes on the game

When we introduced only snakes on the game, the hypothesis was the average game length increases as number of snakes increases. We started from zero snake to twenty snakes the length of the game increased exponentially (see Figure 10). Recall, we use the dice size of 6; however when we fixed the length of snakes at 5 and 10. The game length increased in both but the rate of increase is not as much as random length of the snakes. The Figure 11 shows that the game is shorter when the length of the snakes is 5 compared to when the snakes' length is 10

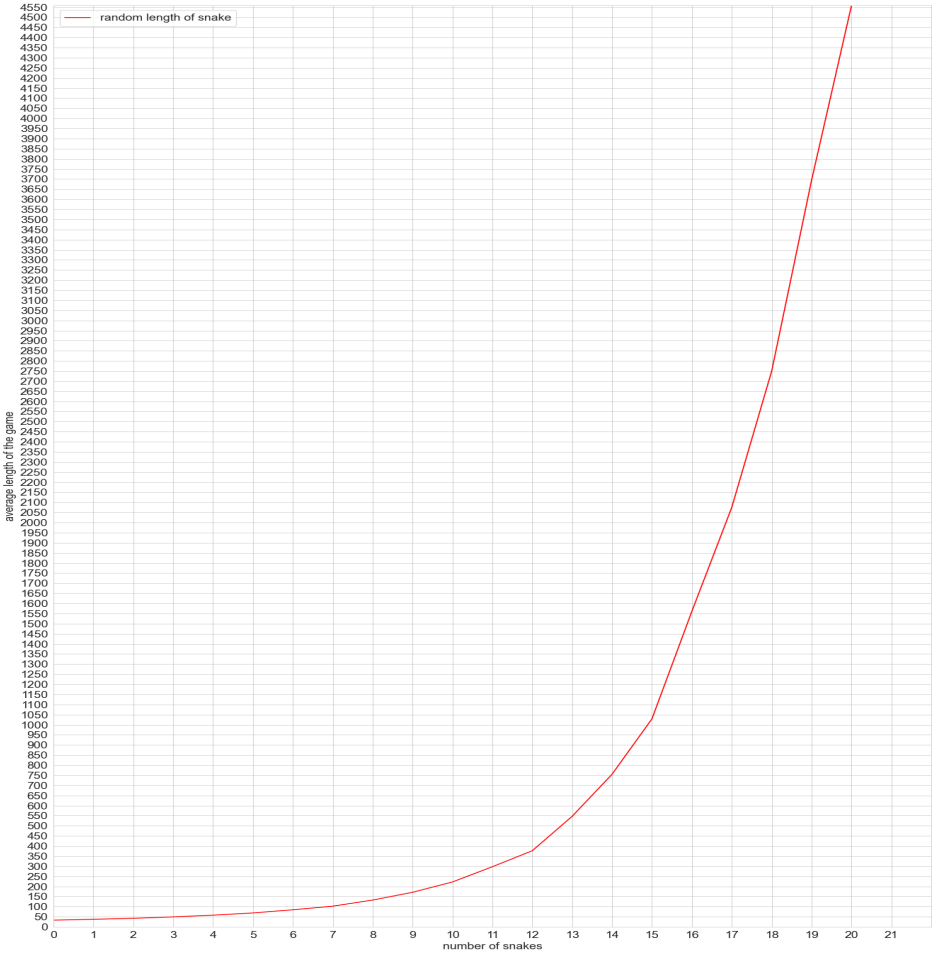


Figure 10: average length of the game with snakes only

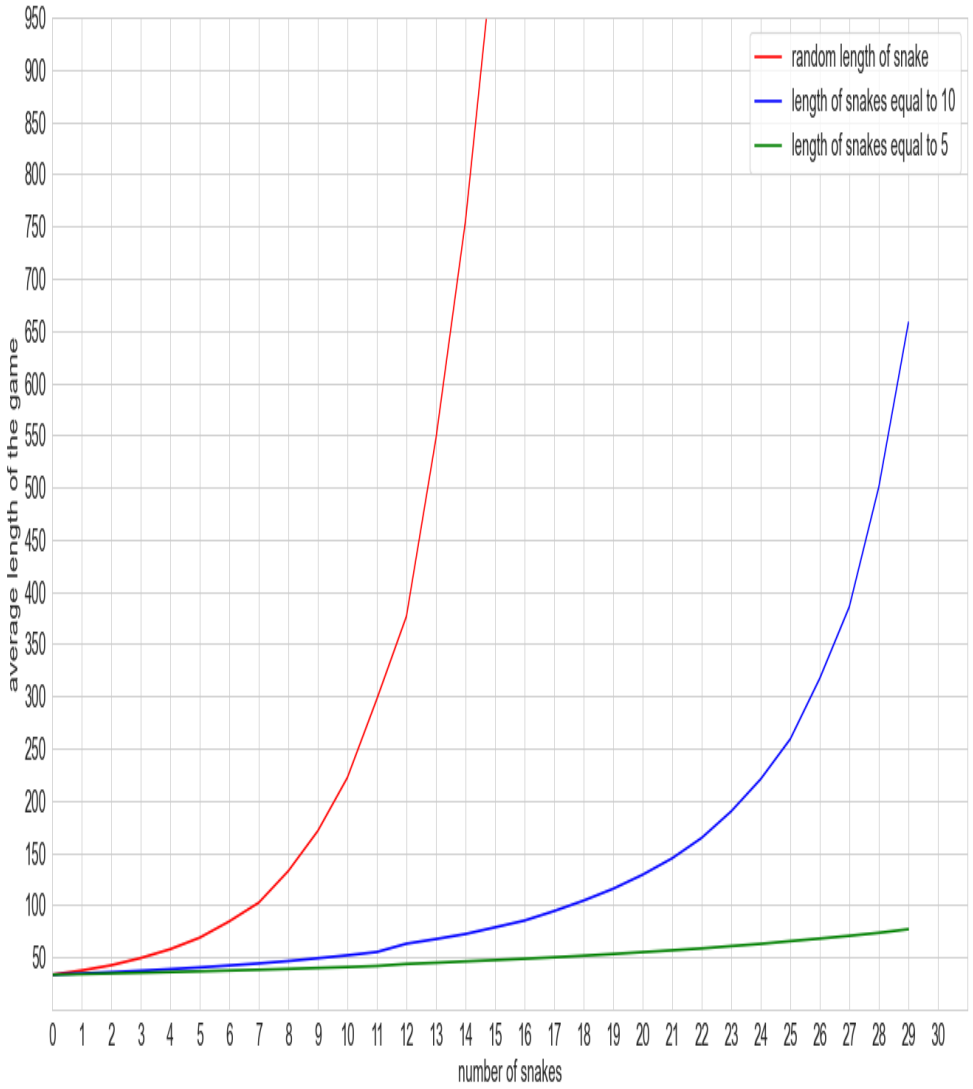


Figure 11: Average length of the game with different length of the snakes

4.1.2 Influence of number of Ladders on the game

We did the same above experiment while keeping the distribution of the die constant but varying number of ladders and length of the ladder only. As expected the length of the game decreases as number of ladders increases and random length of the ladders shorten the game more than when the length of the ladders is 5 or 10.

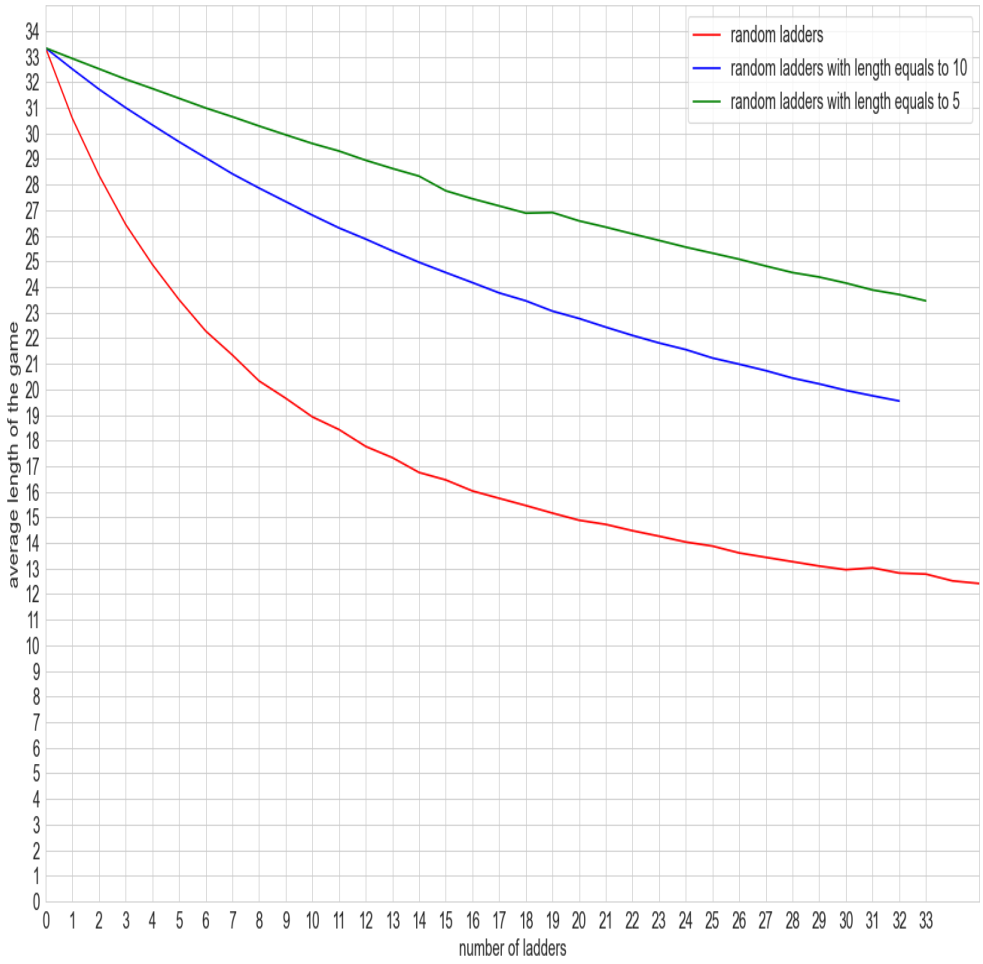


Figure 12: Average length of the game with different length of the ladders

However, the average length of the game can never go below 6 with any distribution of ladders in the game. We found out that if the game has six consecutive ladders from the first to sixth square ascending to the last six squares, i.e., from the hundredth to ninety-sixth square in any arrangement, the average length becomes 6.

4.1.3 Equal number of snakes and ladders

In the Figure 13, the vertical axis represents the average length of the game, and the horizontal axis stands for the equal number of snakes and ladders. The *hypothesis* was that since we add equal number of ladders and snakes the expected average length should be relatively constant .

However, it shows that as the number of snakes and ladders increases, the average length of the game increase. The game was the longest(56) when we had 17 snakes and 17 ladders. Above 10 snakes and ladders except 14, even number of snakes and ladders produced a shorter game length than the previous odd

number of snakes and ladders.For example the average length of the game with 15, 17 and 19 snakes and ladders was 51, 56 and 54.2 respectively, while with 16, 18 and 20 the average length is 49.3, 51 and 50.5 respectively.

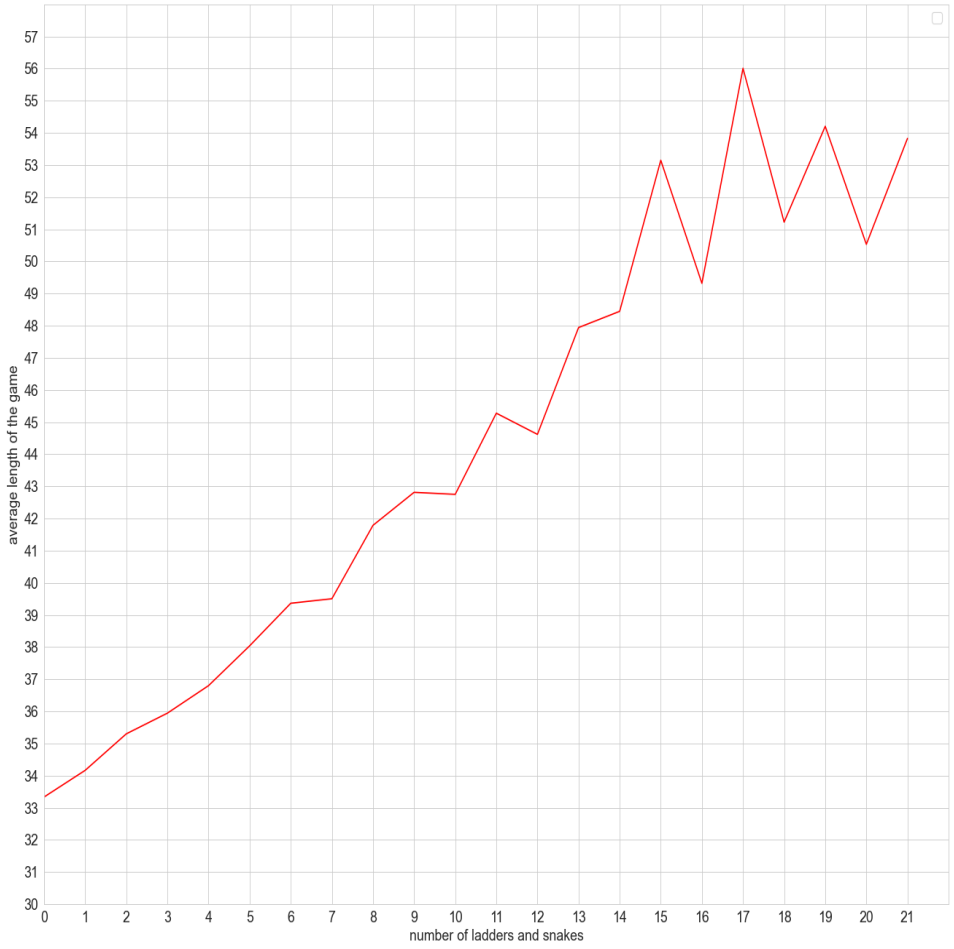


Figure 13: Average length of the game with equal number of snakes and ladders

When we fixed the length of the ladders and snakes at 10 and 5, because we wanted to know if the size of ladders and snakes affect the expected average length of the game. The game with shorter length of ladders and snakes resulted into shorter expected average length. Also, the length of the game increased as number of snakes and ladders increased.

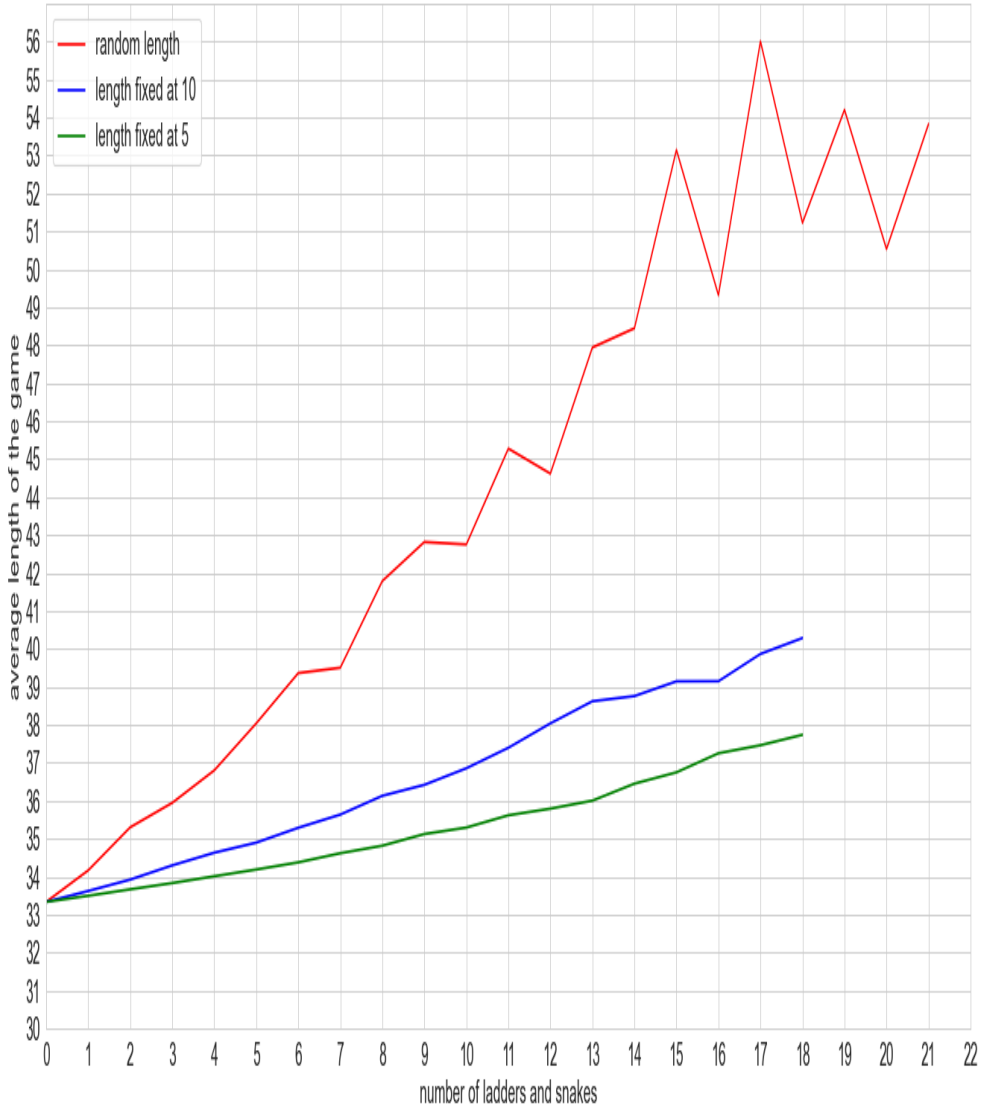


Figure 14: Average length of the game with equal number of snakes and ladders, with fixed length at 5 and 10

4.2 Effect of size of the die

Then, we studied the effect of the die size on the length of the game. We took the die of size 6, 15 and 100. The reason to pick die of size 15 and 100 is because of their peculiar characteristics. For example, a 15-sides die had the shortest average length of the game [LJ11] which was 25.81, in the case of 100-sides die, it gave the most longest game length, which was 90.28 [LJ11]. We did not want to test for 99-sides die even though it also give the longest length of the game 90.28 similar to 100-sides die because article [CG14] has already proven that for the board of size n then die of size n and $n-1$ has same average length

regardless any distribution of snakes and ladders.

4.2.1 Effects of different die's size to the random length of ladders and snakes

We used the same set of snakes and ladders used in Figure 13 however; we used the die of size 6, 15 and 100. As it was said in [LJ11] 15-side die had the shortest game length with a minimum of 26.56 when the game had one snake and ladder, and the maximum was 32.89 when the game had 21 snakes and ladders. In contrast, the 100-side die has a different result; it had a relatively high average game length. Unlike 6-sides or 15-sides die, the 100-sides die game length decreased as the number of snakes and ladders increased. The longest average length was 99.93 when the game had one snake and one ladder; and the shortest was 95.106 when the game had 21 snakes and 21 ladders.

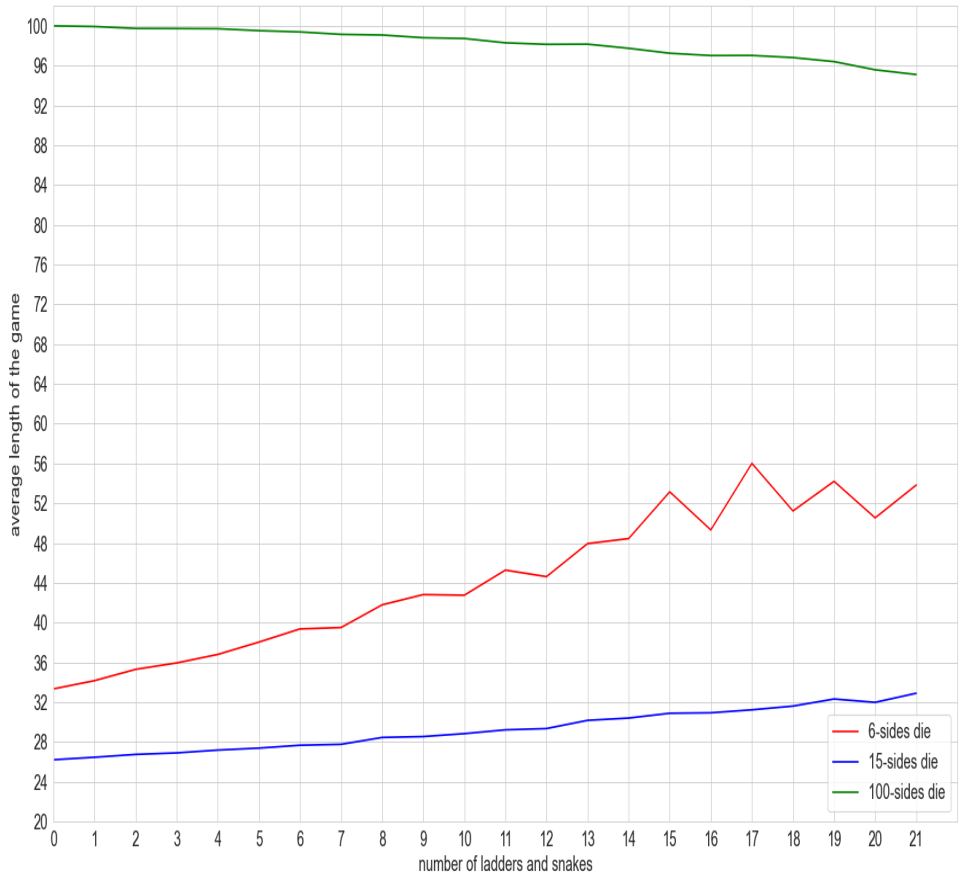


Figure 15: Average length of the game with 6-side,15-side and 100-side die

4.2.2 Effects of different die's size on the game with snakes only

We tested 6-side,15-side and 100-side die on a game that contains only snakes. The Figure 16, shows that the number of snakes increases the game length .

The average game length increases exponentially for 6-sides and 15-sides dice. However, for 100-sides die, snakes had an insignificant effect on the game's length since the game length was 99.9999 for all number snakes and the hypothesis was that 100-side should have the highest average game length. In addition, 6-sides and 15-sides dice passes the average length of 100-sides die when the game had 7 snakes and 21 snakes, respectively. Nevertheless, 15-sides die outperformed 6-sides in any distribution of the snakes.

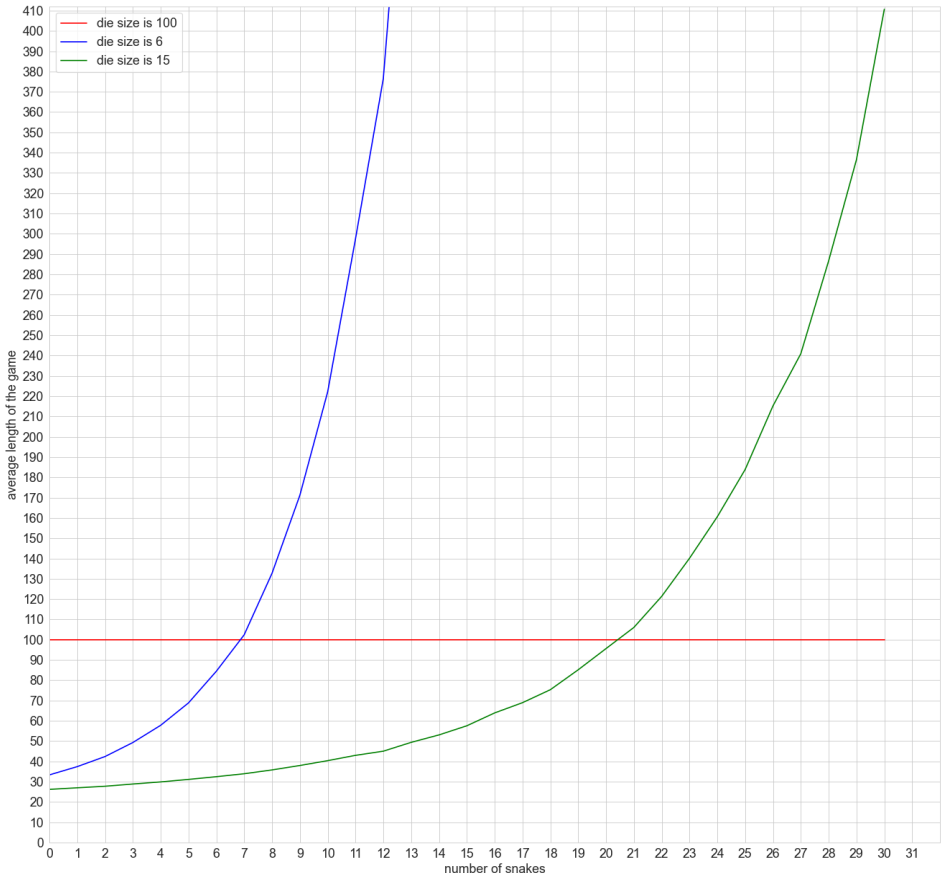


Figure 16: Average length of the game with 6-side,15-side and 100-side die

4.2.3 Effects of different die's size on the game with ladders only

When we used ladders only the average length was decreasing as number of ladders were increasing. Nonetheless, 6-side die outperformed 15-side die as number of ladders increased from when the game had 6 ladders and more. But, the average length of the game was high when we used 100-side die for example the highest was 99.9 when the game had 1 ladder and 96.41 when the game had 48 ladders. This is shown in the Figure 17

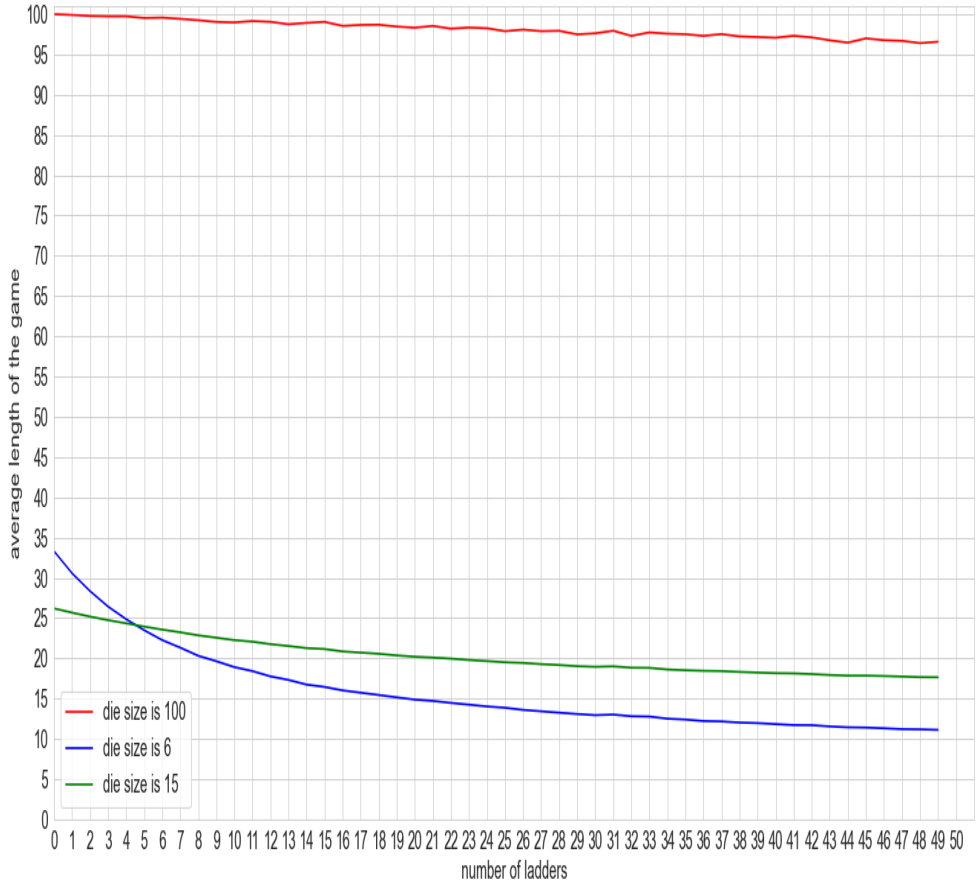


Figure 17: Average length of the game with 6-side,15-side and 100-side die

4.2.4 Length of the game if game ends when token reaches 100^{th} square or above

We changed the rule so that the game can end whenever the token reached the last square or exceeded it. We used the equally distributed 6-sides die. The new rule shows that the average length of the game is shorter compared to the original rule. However, it shows that the game followed the same trend as the game with the original rule. In addition, these two plot lines are not parallel even though they follow the same pattern, and the gap between them widens as the number of ladders, and snakes increases see the table below 2 and Figure 18.

N	original rule	new rule
1	34.163	29.612
2	35.299	30.414
3	35.943	30.774
4	36.798	31.118
5	38.0378	31.9997
6	39.3623	32.939
7	39.503	32.779
8	41.787	33.921
9	42.815	34.872
10	42.751	34.271
11	45.277	36.296
12	44.618	35.489
13	47.942	37.477
14	48.447	37.703
15	53.148	40.225
16	49.318	37.873
17	56.009	41.278
18	51.224	38.296
19	54.205	39.491
20	50.532	38.025
21	53.827	39.812

Table 2: Expected average length for game which ends at exactly 100 square and the game which ends at 100 square or above as a function of number of ladders and snakes N

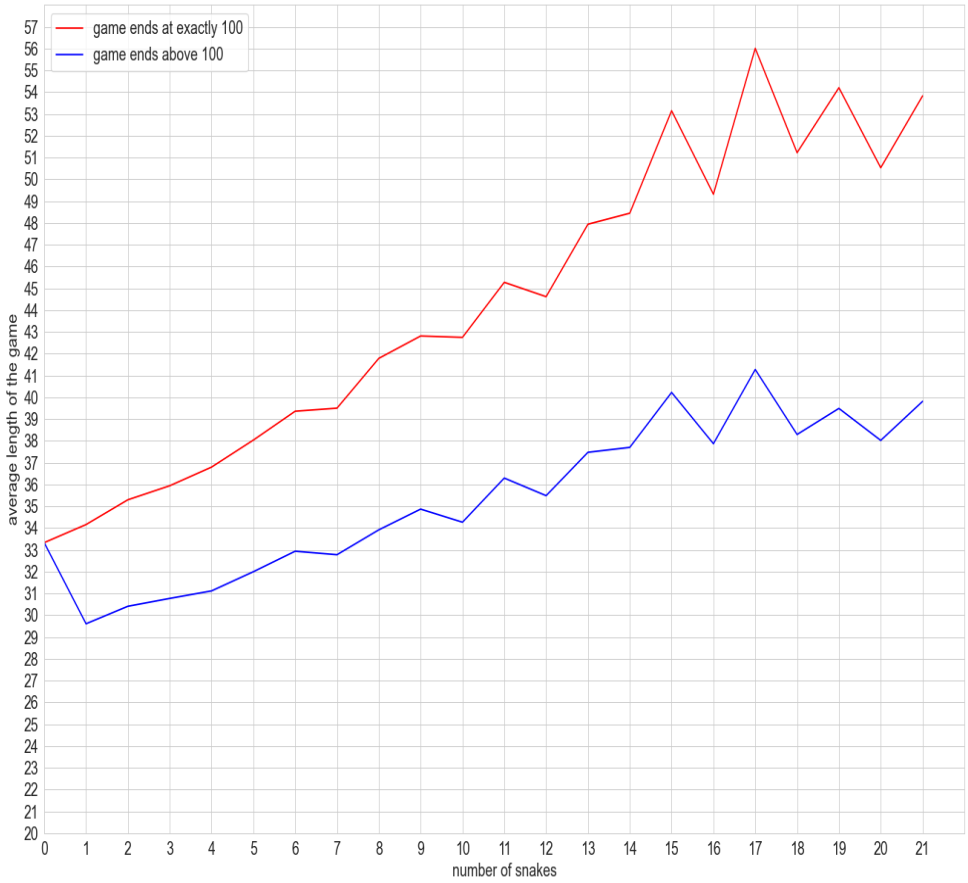


Figure 18: Length of the game if, game ends when token reaches 100^{th} or above

4.2.5 Effect of a pair of dice on the average length of the game

We introduced pair of dice to the game. Recall that we had to change the rule to avoid staying at square 99 forever if the token falls there. Hence, we made one die starting from 0 to 5 and the other die from 1 to 6. The game with pair of dice performed overall well as the number of snakes and ladders increase compared to a single die. Nevertheless, when the game had less than five snakes and ladders, one die had a shorter game length than pair of dice.

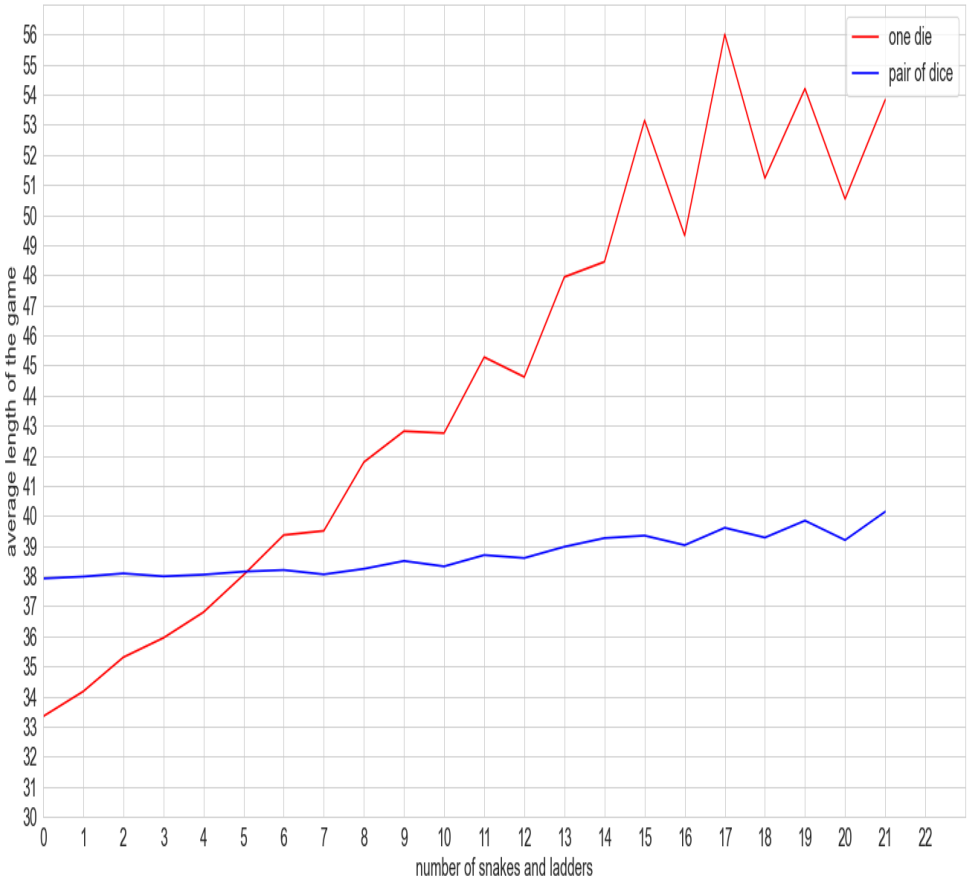


Figure 19: Expected average length of the game with pair of dice

Furthermore, we analyzed the game with 11-sides die and compare them with two dice. The reason to use 11 sides dice is to have equal the jump of token since the maximum jump for pair of dice is 11, but according to this thesis rules, since one die starts from 0. For the game with 21 snakes and 21, a pair of dice gave a longer average game length than 11 side dice. However, the rate of increase of average game length is much higher in 11 sides dice; hence as snakes and ladders increase 11-sides dice will have longer average game length compared to pair of dice. (*see Figure 20*).

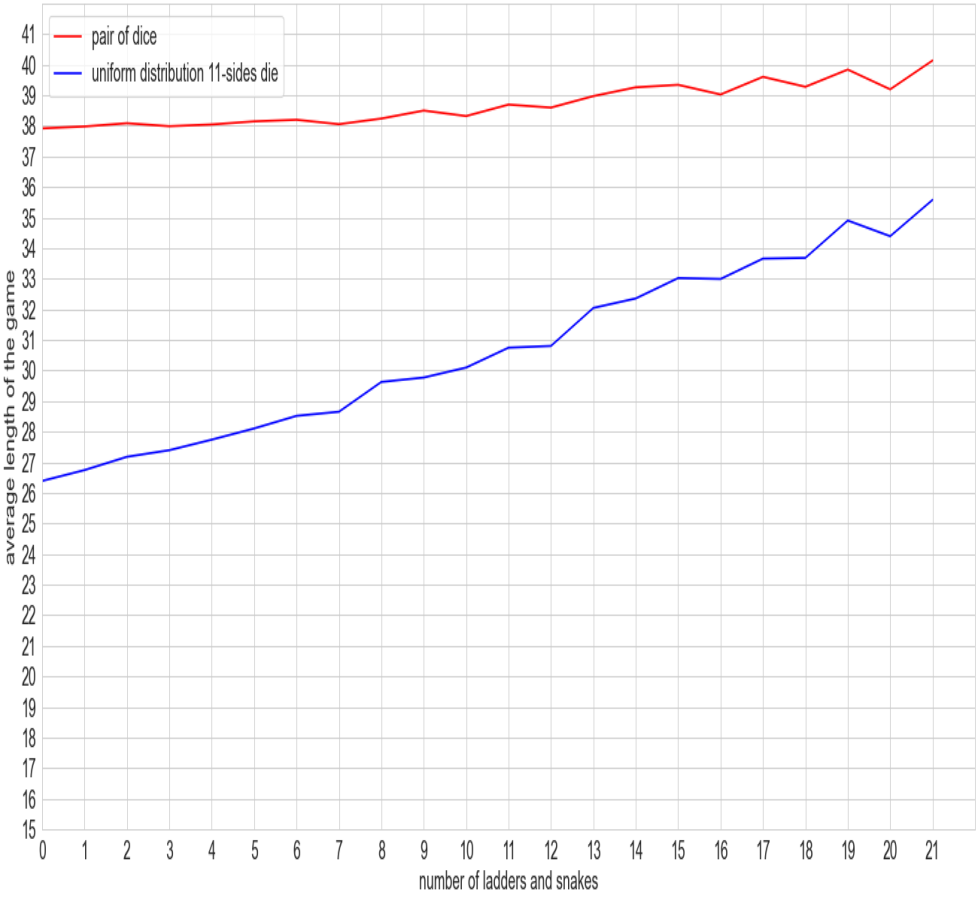


Figure 20: Expected average length for 11-side and pair of dice

4.2.5.1 Effect of a pair of dice on the average length of the game with ladders only

When we studied the game with ladders only, a pair of dice had the highest game length compared to 11-sides. In addition, the shortest average game length of the game with pair of dice, which was 25.95 was larger than 11-sides which was 25.46.

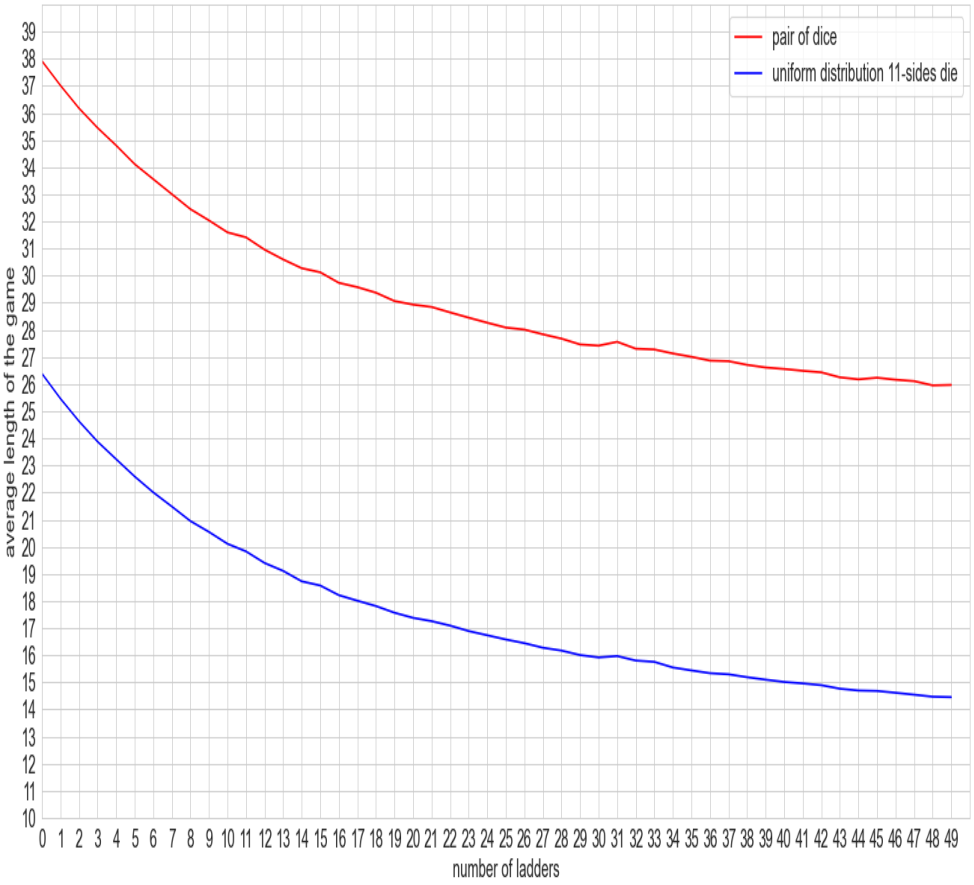


Figure 21: Expected average length for 11-side and pair of dice in a game which has ladders only

4.2.5.2 Effect of a pair of dice on the average length of the game with snakes only

Then we analysed the game with snakes only using 11-sides and pair of dice, as expected the average length of the game increased as number of snakes increased. However, the difference of expected average game length when 11-sides die and pair of dice were used was insignificant. For example when the game had 24 snakes the expected average game length for 11-sides die and pair of dice were 536.99 and 537.37 respectively (*see Figure 22*)

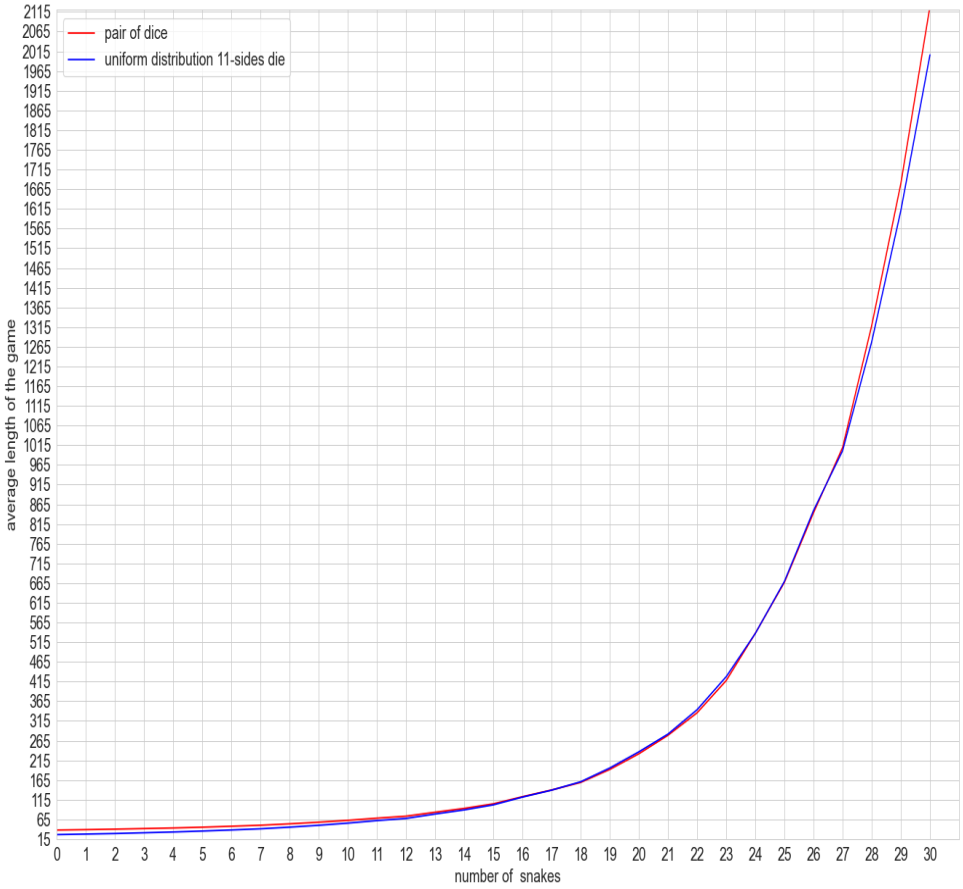


Figure 22: Expected average length for 11-side and pair of dice in a game which has snakes only

4.3 Harmonic distribution die

When answering [LJ11], to find the expected average length of the game using another type of die distribution, we used a harmonic distributed die, recall subsection 3.5. Hence, we used 100-sides die, which is harmonically distributed; the die achieved a shorter average length of the game than a uniform distributed 100-sides die. Recall that we used 1000 sets of snakes and ladders for each number of snakes and ladders. With 21 snakes and 21 ladders, the expected average length was 38.105 when using 100-sides harmonic die, with an exact number of snakes and ladders 100-side uniform die had an average length of 95.106. In addition, unlike uniform 100-sides die where average length was decreasing as the number of ladders and snakes were increasing, the average length of the harmonic 100-sides die was increasing as the number of ladders and snakes were increasing.

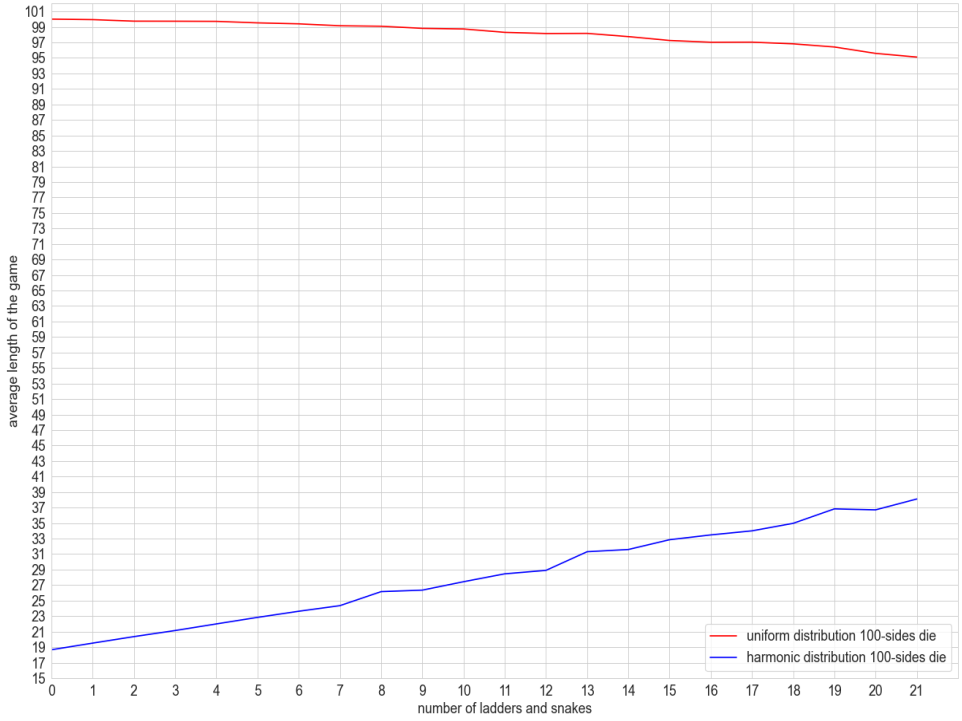


Figure 23: Expected average length for uniform distributed 100-side die and harmonic distributed 100-side die

Then, we varied the size of the harmonic distributed die for the original game. The goal was to find which size of the die gives the shortest length of the game as demonstrated in article [LJ11] which used uniform distributed die (see table 1). The table 3 shows the size of the dice against the expected duration to complete the game. The minimum expected average occurred when the game was played with 69-sides die, which was 21.755. From die of size 3 to 68 the expected average length decreased, and from 70 to 99 the game length slightly increased; notice that the expected average length at die of size 99 and 100 is equal [CG14].

n	E	n	E	n	E	n	E	n	E
1	∞	21	27.572	41	22.625	61	21.765	81	21.814
2	69.871	22	26.983	42	22.533	62	21.761	82	21.843
3	71.016	23	26.538	43	22.479	63	21.768	83	21.872
4	63.664	24	26.104	44	22.426	64	21.775	84	21.903
5	56.767	25	25.650	45	22.359	65	21.768	85	21.928
6	51.483	26	25.257	46	22.290	66	21.760	86	21.952
7	47.106	27	24.884	47	22.235	67	21.759	87	21.988
8	44.078	28	24.606	48	22.192	68	21.758	88	22.011
9	41.766	29	24.417	49	22.167	69	21.755	89	22.034
10	39.528	30	24.209	50	22.124	70	21.758	90	22.062
11	37.544	31	24.017	51	22.078	71	21.764	91	22.090
12	35.977	32	23.821	52	22.042	72	21.769	92	22.116
13	34.172	33	23.652	53	21.9995	73	21.766	93	22.142
14	32.965	34	23.490	54	21.945	74	21.764	94	22.164
15	31.732	35	23.350	55	21.9	75	21.763	95	22.194
16	30.790	36	23.196	56	21.847	76	21.764	96	22.217
17	30.014	37	23.069	57	21.825	77	21.778	97	22.236
18	29.407	38	22.957	58	21.804	78	21.793	98	22.263
19	28.793	39	22.832	59	21.793	79	21.811	99	22.273
20	28.272	40	22.715	60	21.783	80	21.788	100	22.273

Table 3: The expected number E of turns for one player to complete Snakes and Ladders game, as a function of spinner or die range n

The Figure 24 shows that uniform die outperforms harmonic die when the size of both dice are below 21; afterwards, the harmonic die has better performance than uniform die. In addition, when the size of both dice is 3 has a higher game length compared to when the size was 2 or 4. Furthermore, when using the above approach for the game with no snakes and ladders (see Figure 25), the game follows the same trend except 3-sides die now has lower average length compared to 2-sides die.

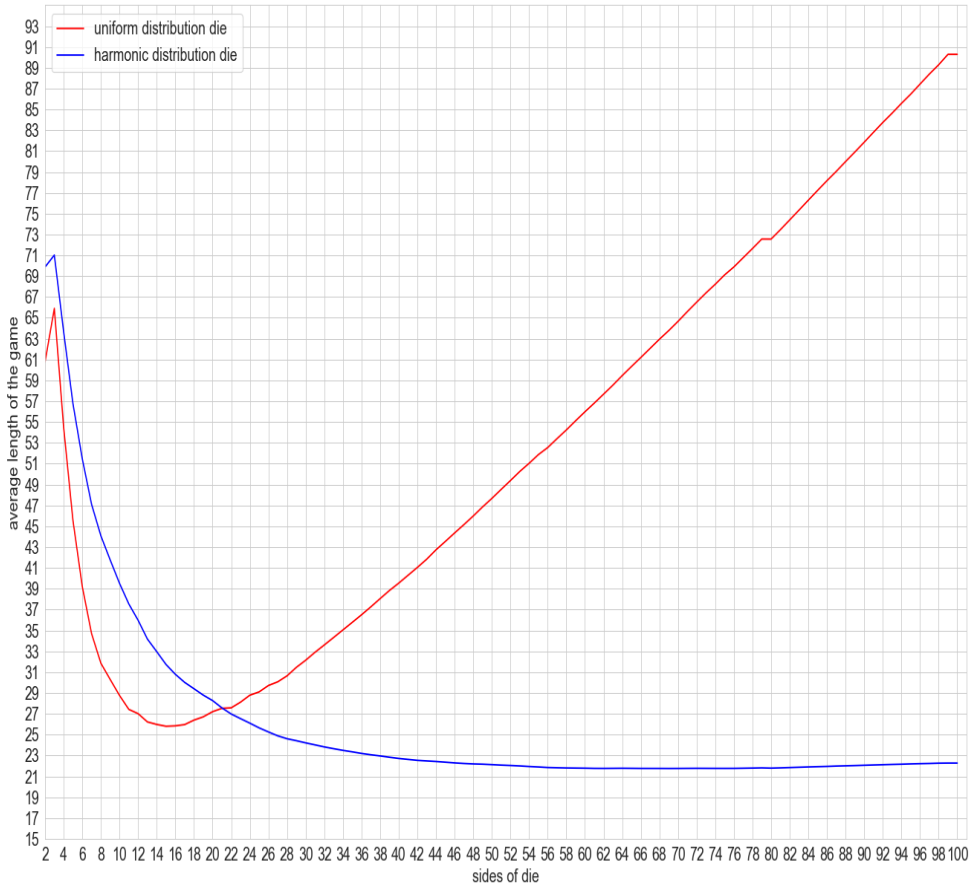


Figure 24: Expected average length for uniform distributed 100-sides die and harmonic distributed 100-sides die with original set of snakes and ladders

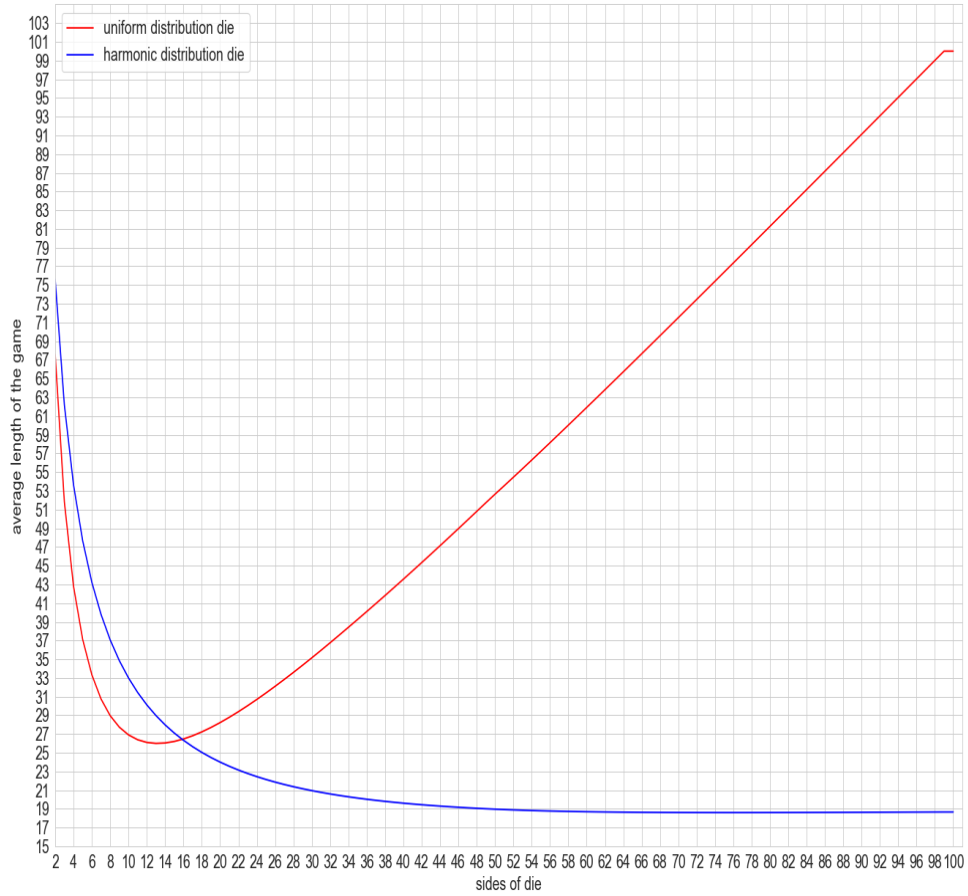


Figure 25: Expected average length for uniform distributed 100-sides die and harmonic distributed 100-sides die without snakes and ladders

When we compared uniform 15-sides die with harmonic 100-side die in the game with random ladders and snakes. The 15-sides uniform die initially had a higher expected average length than the 100-sides harmonic die. However, when the number of snakes and ladders were above 13; the 15-sides uniform die had lower expected average length compared to the 100-sides harmonic die (see Figure 26).

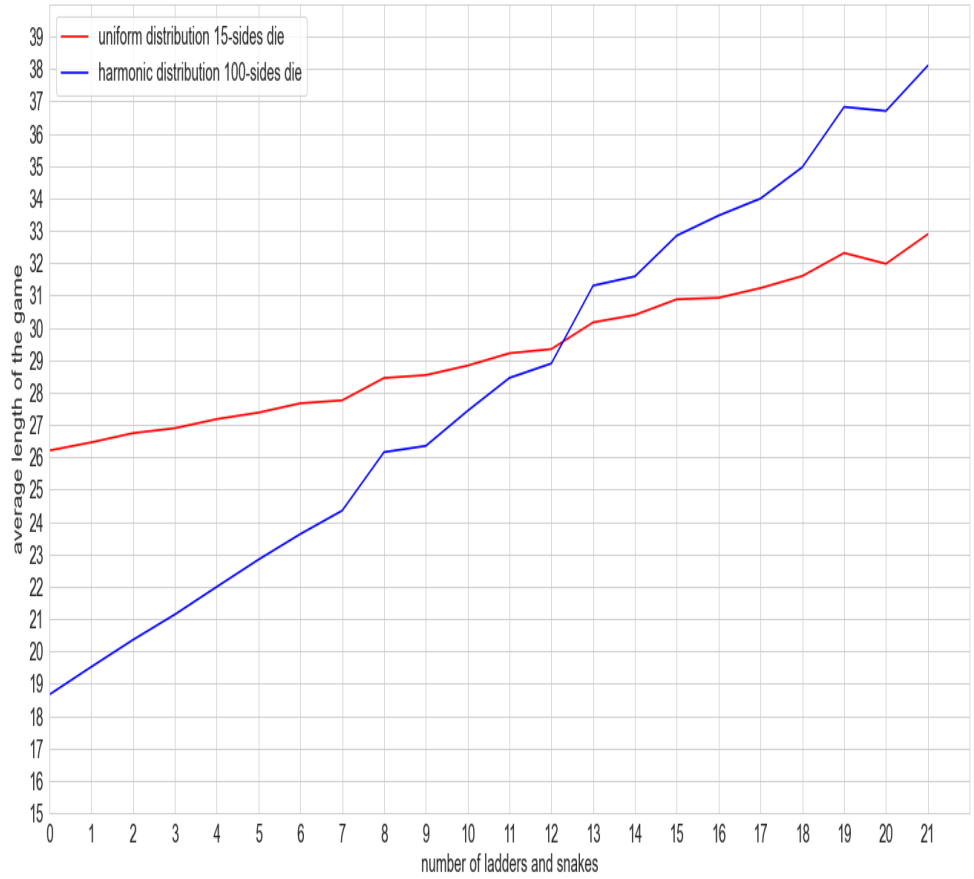


Figure 26: Expected average length for uniform distributed 15-sides die and harmonic distributed 100-sides die

We experimented 100-side harmonic die in the game with only snakes. Unlike 100-sides uniform die where snakes did not affect the game length, which was 99.9999, for harmonic die, average game length increased as the number of snakes increased (see Figure 27).

In contrast, when analyzing the game with ladders only using 100-sides harmonic die, the average game length decreases as the number of ladders increases but at a slower rate. The lowest average game length was when the game had 49 ladders. In addition, 100-side harmonic die achieved lower average game length compared to 100-side uniform die (see Figure 28).

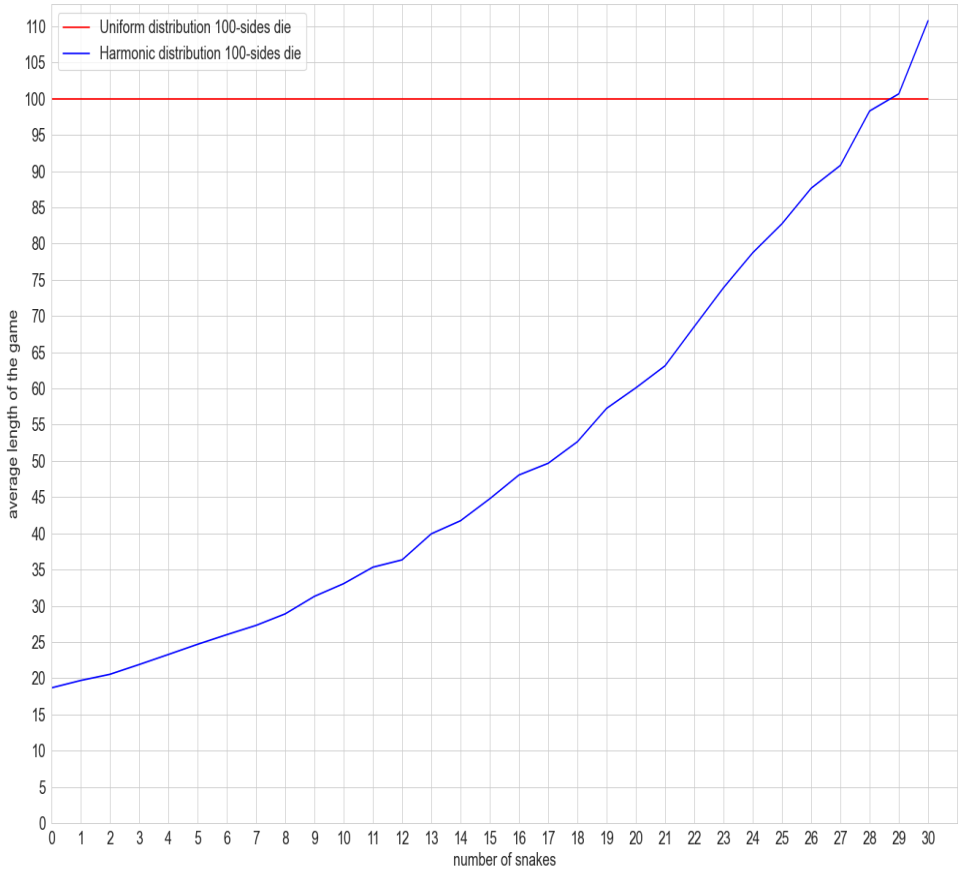


Figure 27: Expected average length for uniform distributed 100-sides die and harmonic distributed 100-side die with only snakes

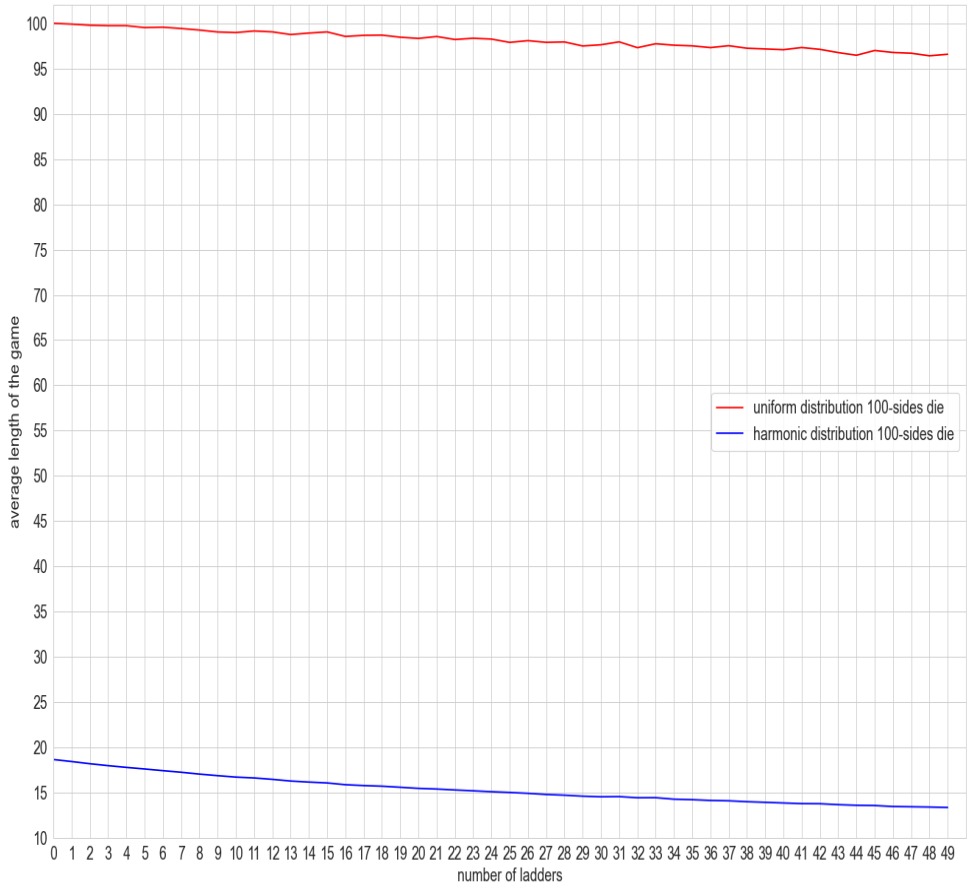


Figure 28: Expected average length for uniform distributed 100-side die and harmonic distributed 100-sides die with ladders only

Furthermore, we analyzed the game using 100-sides and 69-harmonic harmonic dice because we wanted to know the average times to visit a square in the game. We also wanted to know which are the least and most visited squares with both dice. When we used 100-sides harmonic die, 99 had the most visits, which was 1.829 per game (see Figure 29). In addition the top five most visited squares are $99, 97, 78, 0$ and 91; and the least visited squares are $49, 93, 87, 80$ and 71. For 69-sides harmonic die 99 was the most visited square approximately 1.684 times. Also , the most five visited square were $99, 97, 0, 78$ and 91 and least visited were $49, 98, 51, 36$ and 28.

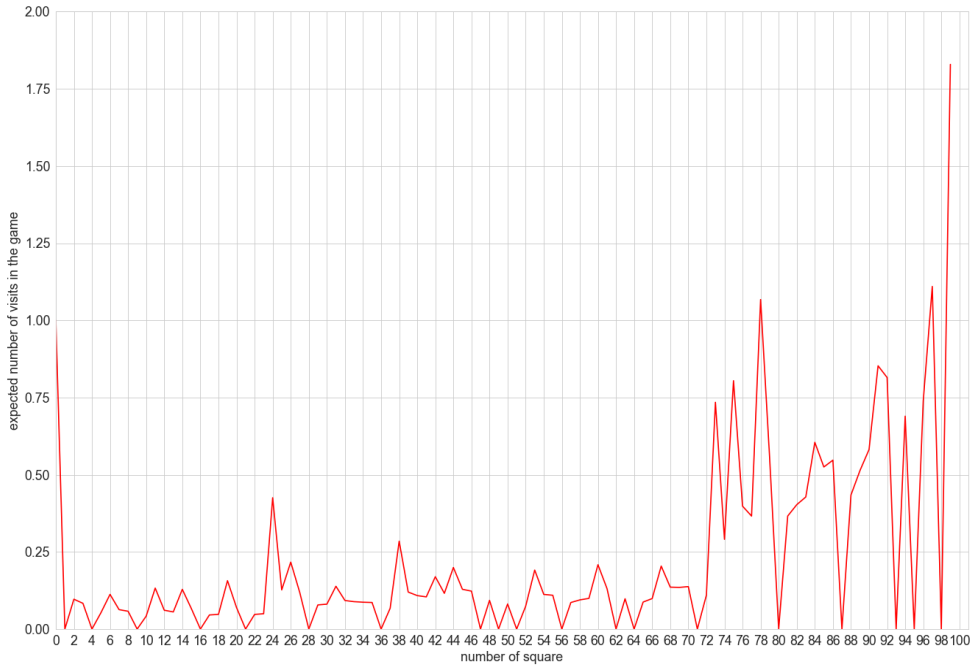


Figure 29: Expected number of square visits in the game with 100-sides harmonic die

5 Discussion

5.1 Effects of number of snakes and ladders in the game

The number of ladders and snakes primarily affects the expected average length of the game. For 6-sides die, we expected that the average length should have been relatively constant as the number of ladders and snakes were increase at the same rate. However, the number of snakes has notable effects compared to the number of ladders, leading to a higher average game length. This is shown in graph 10 whereby when the game had 13 snakes without ladders, the average length was about 539, but with the addition of one more snake, the game length was 739. Hence game length increased by 200, while for the game, 13 ladders without snakes, the average length was 16.8, and with the addition of one ladder, the average length became 16.5. Therefore, snakes affect the length of the game much more than ladders.

Also, the length of the snakes ladders and snakes have an impact on the average game length as we have seen that snakes with a shorter length lead to lower game length 11. In graph 14 the game had the lowest average game length when the length of snakes and ladders were fixed at 5. The fig. 12, fig. 11 and Figure 14 also support the claim that snakes have much influence on the length of the game than ladders.

Furthermore, the position of the snakes and ladders have significant effects on the length of the game, in [AKS93] noted that it depends on where you put the snake or a ladder. For example, [AKS93] added a snake 29 to 27 to the

original game; the average length was 38.0, which was lower compared to 39.2 of the original game since the new snake gave the token a chance to land to the longest ladder 28 to 84. However, if we maintain the length of the ladders or snakes but changing their position, we found out that ladders that land a token to the final row have a shorter game than the ladder, which lands the token to other rows. For example, when we added a ladder from state 81 to state 91 to the original game, the average became 35.22 which was lower than the original game, in contrast when we replaced the previous ladder with a new one from state 23 to 33. The game average length change was insignificant which was 39.86. Furthermore, when we added a new snake from state 92 to 82 in the original game, the average length rose to 43.52, but when we substituted it with a snake from state 22 to 12, the average slightly rose to 40.84. Although the length of the snakes and ladders affect the game length, the position on where they are placed have a significant impact on the average length of the game.

So, when we consider the fixed-length of ladders or snakes. The ladders which make the token land on any state from 90 to 100 lowers the average length, and the ladder which lands the token anywhere below 90 state has little impact on the game length. Unlike ladders, snakes that start from state 90 to 99 increase significantly the average length of the game than the snakes, which start from any state below 90.

5.2 Influence of Dice distribution on the game length

The other parameter we studied in this thesis is the different distribution of a die. We analyzed the effect of die distribution on the average game length. We studied 12-sides, 11-sides, 6-sides(original die), 15-sides and 100-sides uniform dice; furthermore we studied 100-sides harmonic die and pair of uniform 6-sides dice.

Before going further into die distribution, we must mention that when we changed the rule, that game ends as long as a token passes 100 state. The new rule lowers the average length of the game significantly (see graph 18). For example, the average length for this new rule is 35.83 which is less than the original average game length

When we analyzed a pair of 6-sides uniform dice, we found out that pair of dice performed very well against the 6-side uniform die. However, recall that we set one of the dice to start from 0 to 5 hence avoiding the game to be stuck at 99 state (*original rule games ends when a token falls strictly at 100 square*) since the minimum token jump will be 2 steps.

We then studied 11-sides and 12-sides dice; we found that 11 and 12-sides dice perform better than pairs of dice throughout the experiment. The reason to choose 11-sides and 12-sides was to maintain steps jump between pair of dice and uniform die, because pair of dice has a total of 12 jumps in our case, 11 jumps since one die starts from 0 to 5. However, when we used the game with snakes, only the average length for pair of dice and 11-sides dice was slightly similar when the game had 12 snakes to when the game had 27 snakes; nevertheless, the 12-sides die outperform both with relatively lower average game length.

Article [LJ11] pointed out that 15-sides die gave optimal lowest average game length in the original game. Hence, in this thesis, we tried to determine if 15-sides die performs best in any distribution of snakes and ladders. We found out that the 15-sides uniform die gives the lowest average game length in any

distribution of snakes and ladders compared to any other n-sides uniform die (*see graph 15*).

Then we experimented 100-sides and 15-sides uniform dice in the game, which contains snakes only. For 15-sides, the average game length increased exponentially as the number of snakes increased but at a slower rate compared to 6-sides die. For 100-sides uniform die, the results were quite different. The average game length remained constant at 99.9999 in any distribution of snakes in the game. When we tried a game with ladders only, 6-sides uniform die outperform both 15-sides and 100-sides uniform die with the lowest average length. This experiment showed that snakes only influence the game length on other die distribution except 100-sides uniform die; for 100-sides die, only ladders influence the average length.

Furthermore, we used the harmonic die to find its performance in the game. Article [Die+10] pointed that harmonic distribution spinner give average length of $\mathbf{O}((\mathbf{log}\ n)^2)$ which is the minimum for any discrete token process. However, to achieve this average length, the size of the die or spinner should be equal to the number of states in that process. On the contrary, we found out that the 100-sides harmonic die does not give the optimal shortest average length. The 69-sides die gives the optimal minimum average game length, which is 21.755 in the original game, and when we used the game with no snakes and ladders, the minimum average length of 18.598 was given by the 76-sides harmonic die.

6 Conclusion

This thesis aimed to analyze the average length of the game of snakes and ladders by changing different parameters of the game such as die distribution, length and number of the ladders and snakes. Based on the results, we can conclude that number of snakes has a significant influence on the average length of the game. However, for 100-sides uniform die, the number of snakes does not affect the average length of the game.

Also, the distribution of a die has a massive impact on the game length. For example, the game with less than 12 snakes and 12 ladders 100-sides harmonic gives a lower average game length than any size of the uniform die. However, 15-sides uniform die gives a lower average game length when ladders and snakes are more than 13.

In addition, we proved that die of size *n and (n-1)* give the same expected average length regardless of die' distribution (in our case *harmonic distribution*) used in the game [CG14].

In contrary to the article [Die+10], we found out that to achieve minimum average game length, the size of the die should not necessarily be equal to the number of states in the Markov chain.

In conclusion, 15-sides uniform die gives the lowest average length among other distributions of a die; snakes significantly influence the average game length compared to the ladders. The position of the snakes and ladders also have a notable influence on the average length of the game.

6.1 Recommendation

This thesis used Mersenne Twister as a pseudo-random number generator (PRNG) to simulate the game. In the original game, the average length of a simulation was approximately equal to the theoretical value from Markov chain analysis when a uniform die was used. For example,in the original game, the average length from the simulation was 39.23276, and the average from Markov chain analysis was 39.225. However, when we simulated the game using the harmonic die, the simulated average was much different from Markov analysis, when the number of sides of the die is greater than 8. For example, optimal minimum average length from simulation was 26.804 with 41-sides die while Markov analysis gave 21.755 with 69-sides die (see Table 3 and Figure 30).
Recommendations:

- study on why Mersenne Twister give significant difference from Markov analysis when the size of the harmonic die is greater than 8. *See Figure 30 and Figure 9*
- finding another distribution of the die which will shorten the game length

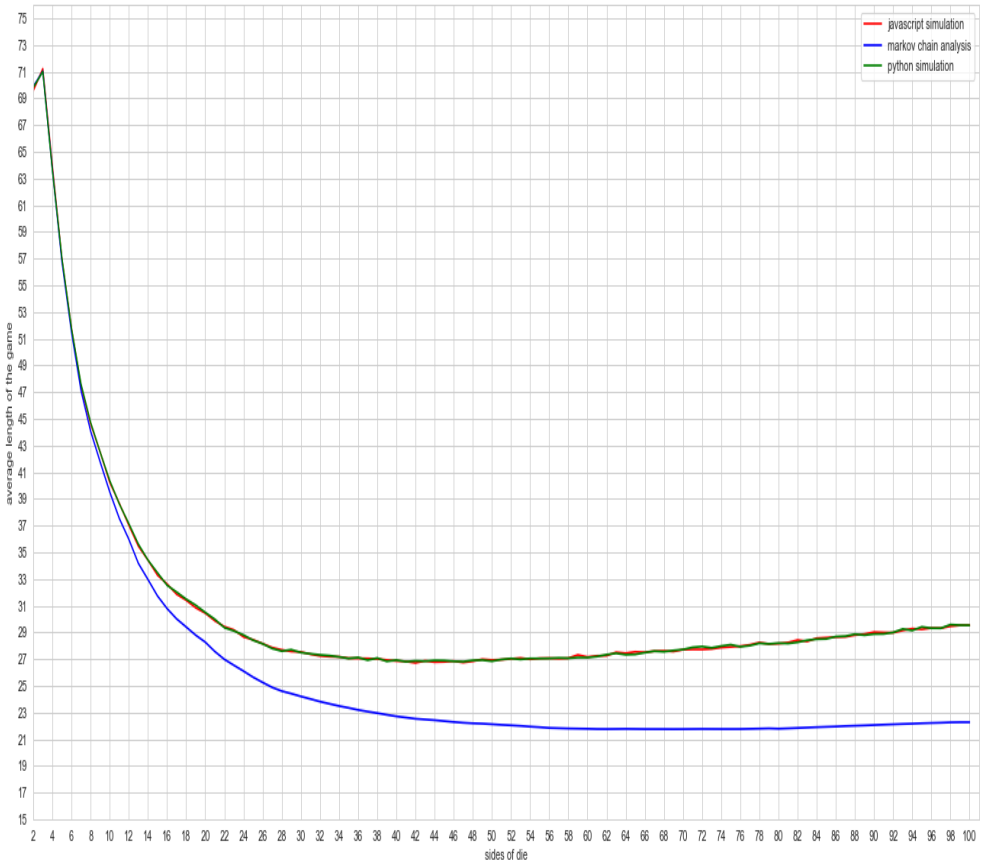


Figure 30: Average game length for simulation and Markov chain analysis using harmonic die

All the works in this thesis are found in Github, visit this link `https://github.com/tonnyhideyori/algo_thesis`

References

[Usp37] J.V. Uspensky. *Introduction to mathematical probability*. McGraw-Hill Book Company, 1937.

[Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*. Vol. 1. Wiley, Jan. 1968. ISBN: 0471257087.

[AKS93] S. C. Althoen, L. King, and K. Schilling. "How Long Is a Game of Snakes and Ladders?" In: *The Mathematical Gazette* 77.478 (1993), pp. 71–76. ISSN: 00255572. URL: http://www.jstor.org/stable/3619261.

[Joh03] Roger W. Johnson. "Using games to teach Markov chains". In: *PRIMUS* 13 (4 Jan. 2003). DOI: 10.1080/10511970308984067.

[GS06] Charles Miller Grinstead and James Laurie Snell. *Introduction to probability*. American Mathematical Society, 2006.

[Die+10] Martin Dietzfelbinger, Jonathan E. Rowe, Ingo Wegener, and Philipp Woelfel. "Tight Bounds for Blind Search on the Integers and the Reals". In: *Combinatorics, Probability and Computing* 19.5-6 (2010), pp. 711–728. ISSN: 0963-5483. DOI: 10.1017/s0963548309990599.

[LJ11] Stewart Hengeveld Leslie A. Cheteyan and Michael A. Jones. "Chutes and Ladders for the Impatient". In: *College Mathematics Journal* 42 (1 Jan. 2011). DOI: 10.4169/college.math.j.42.1.002.

[CG14] Darcie Connors and Darren Glass. "Chutes and Ladders with Large Spinners". In: *The College Mathematics Journal* 45.4 (2014), pp. 289–295. ISSN: 0746-8342. DOI: 10.4169/college.math.j.45.4.289.

[DB15] Lokenath Debnath and Kanadpriya Basu. "A short history of probability theory and its applications". In: *International Journal of Mathematical Education* 46 (1 Jan. 2015). DOI: 10.1080/0020739x.2014.936975.

[Poi19] S. -D. Poisson. *English Translation of Poisson's Recherches sur la probabilité des jugements en matière criminelle et en matière civile (Researches into the Probabilities of Judgements in Criminal and Civil Cases)*. 2019. arXiv: 1902.02782 [math.HO].

[Wei] Eric W. Weisstein. *Dice*. Ed. by MathWorld. URL: https://mathworld.wolfram.com/Dice.html.

A Appendix

A.1

```
import numpy as np
total = []

def walks():
    stop = "U"
    count = 0
    path = []
    while stop != "B":
        if stop == "U":
            stop = "K"
            count += 1
            path.append(stop)
        if stop == "K":
            stop = np.random.choice(["BS", "S"], size=1, replace=
    True, p=[0.7, 0.3])[0]
            count += 1
            path.append(stop)
        if stop == "S":
            stop = np.random.choice(["K", "BS"], size=1, replace=
    True, p=[0.20, 0.8])[0]
            count += 1
            path.append(stop)
        if stop == "BS":
            stop = np.random.choice(["K", "S", "B"], size=1,
    replace=True, p=[0.05, 0.15, 0.80])[0]
            count += 1
            path.append(stop)
    return [count, path]


for i in range(0, 100000):
    total.append(walks())
sums = 0
#find the average
for ele in total:
    sums += ele[0]
print(f'average walk is {sums/len(total)}')
```

Listing 1: random walk simulation

```
import numpy as np
#initialization of parameters
board = 100
dice=6
state = board+1
snakes_ladders = [[1, 38], [4, 14], [9, 31], [21, 42], [28, 84],
    [36, 44], [51, 67], [71, 91], [80, 100], [16, 6], [47, 26],
    [49, 11], [56, 53], [62, 19],[64, 60], [87, 24], [93, 73], [95,
     75], [98, 78]]

# no snake or ladder transition matrix
def transition_matrix1(dice, board, state):
    M = np.zeros((state, state), dtype=float)
    for i in range(0, state):
        for j in range(1, state):
            if j <= dice and i <= board-dice:
                M[i, j+i] = 1/dice
```

```
            if i >= board-dice:
                M[i, i] = ((dice-board)+i)/dice
                if j+i < len(M):
                    M[i, j+i] = 1/dice
    return M

# transition matrix with  snake and ladder
def snake_ladder(self, Y, snakes):
        for i in range(0, self.state):
            for j in range(0, self.state):
                for snake in snakes:
                    if j == snake[0]:
                        x = Y[i, snake[0]]
                        Y[snake[0]] = 0
                        Y[i, snake[1]] = Y[i, snake[1]] + x
                        Y[i, snake[0]] = 0
        return Y

# calculating length of the game based on fundamental form
def fundamental_form(M):
    Q = M[:-1, :-1]
    N = np.linalg.inv(np.identity(len(Q))-Q)
    length = np.matmul(N, np.ones((len(N), 1)))
    return length[0][0]

# number of visits in the game
def number_visit(self, M):
    Q = M[:-1, :-1]
    N = np.linalg.inv(np.identity(len(Q)) - Q)
    return N

#more than 100
def fundamental_form1(self, M):
    Q = M[:-6, :-6]
    N = np.linalg.inv(np.identity(len(Q)) - Q)
    length = np.matmul(N, np.ones((len(N), 1)))
    # print(self.bmatrix(length))
    return length[0][0]

#finding the expected length of the game
def expectation():
    print(fundamental_form(snake_ladder(
            transition_matrix1(dice, board, state), snakes_ladders,
     state)))
```

Listing 2: Markov chain for snakes and ladders game

```
import numpy as np
board = 100
dice=6
state = board+dice+1

snakes_ladders = [[1, 38], [4, 14], [9, 31], [21, 42], [28, 84],
    [36, 44], [51, 67], [71, 91], [80, 100], [16, 6], [47, 26],
    [49, 11], [56, 53], [62, 19],
                  [64, 60], [87, 24], [93, 73], [95, 75], [98, 78]]

# no snake or ladder transition matrix
def transition_matrix1(dice, board, state):
    M = np.zeros((state, state), dtype=float)
    for i in range(0, state):
        for j in range(1, state):
            if j <= dice and i <= board-dice:
```

```
                M[i, j+i] = 1/dice
            if i >= board-dice:
                if i < board  and state-j<=dice:
                    M[i, state-j+i] = 1/dice
            if i>=board:
                M[i,i]=1
    return M

# transition matrix with  snake and ladder
def snake_ladder(M, snakes, state):
    for i in range(0, state):
        for snake in snakes:
            if i < snake[0]:
                M[i, snake[1]] = M[i, snake[1]] + M[i, snake[0]]
            M[snake[0], i] = 0
            M[i, snake[0]] = 0
    return M

#harmonic transition matrix
def transition_harmonic(self):
    acc=0
    M = np.zeros((self.state, self.state), dtype=float)
    for i in range(0, self.state):
        for j in range(1, self.state):
            if j <= (self.dice) and i <= (self.board-self.dice):
                M[i, j+i] = self.dist[j-1]
            if i>self.board - self.dice:
                if j+i<=self.board:
                    M[i,j+i]=self.dist[j-1]
    for i in range(0,self.state):
        for j in range(1,self.state):
            acc += M[i,j]
        M[i,i]=1-acc
        acc=0
    return M

#transition matrix for the game which end as sou as it reaches 100
    and more
def transition_matrix2(self):
    self.state=self.state + self.dice - 1
    M = np.zeros((self.state, self.state), dtype=float)
    for i in range(0, self.state):
        for j in range(1, self.state):
            if j <= self.dice and i <= (self.board - self.dice):
                M[i, j+i] = 1/self.dice
            if i >= (self.board-self.dice):
                if i < self.board  and (self.state-j)<=self.dice:
                    M[i, self.state-j+i] = 1/self.dice
            if i>=self.board:
                M[i,i]=1
    return M

#transition matrix for the game which has pair of dice
def transition_matrix3(self):
    acc=0
    Tdice=2*self.dice
    M=np.zeros((self.state,self.state),dtype=float)
    for i in range(0,self.state):
        for j in range(1,self.state):
            if j <= (Tdice) and i <= (self.board):
                if j+i <= len(M):
                    M[i,i+j-1]=(self.dice - abs(j-(self.dice+1)))/(
```

```
    self.dice*self.dice)
    for i in range(0,self.state):
        for j in range(1,self.state):
            acc += M[i,j]
        M[i,i]=1-acc
        acc=0
    return M

# calculating length of the game based on fundamental form
def fundamental_form(M):
    Q = M[:-(dice+1),:-(dice+1)]
    N = np.linalg.inv(np.identity(len(Q))-Q)
    length = np.matmul(N, np.ones((len(N), 1)))
    return length[0][0]

#finding the expected length of the game
def expectation():
    print(fundamental_form(snake_ladder(
            transition_matrix1(dice, board, state), snakes_ladders,
     state)))
```

Listing 3: Markov chain for snakes and ladders game with rule that game ends as long as it exceeds 100

```
import random
class game(object):
    def __init__(self,dice=6,sl=None):
        self.dice =dice
        self.sl = sl
        self.dist=[]
        self.our=[]
        nth=Harmonic(self.dice)
        for d in range(1,self.dice+1):
            self.dist.append(1/(d*nth.harmonic()))
        if sum(self.dist)>1:
            self.dist[len(self.dist)-1]=self.dist[len(self.dist)
    -1]+(1-sum(self.dist))
            if sum(self.dist)<1:
               self.dist[len(self.dist)-1]=self.dist[len(self.dist)
    -1]+(1-sum(self.dist))
        else:
            self.dist[len(self.dist)-1]=self.dist[len(self.dist)
    -1]+(1-sum(self.dist))

    #simulation of the original snakes and ladders game
    def gamesimulation(self):
    # initialization of parameters count for counting the length of
     game, path records the step of the game to completion, token
    is actual element moving through the game
        count = 0
        path = []
        token = 0
        while token < 100:
            roll = random.randint(1, self.dice)
            token = token + roll
            count += 1
            # contorls token should land exactly at 100
            if token > 100:
                token = token - roll
            for trans in self.sl:
                if token == trans[0]:
                    token = trans[1]
```

```
                    break
            path.append(token)
        return [count, path]

    #simulation for the game where by the end of the game is not
    exactly 100 position but >=100
    def gamesimulation3(self):
        # initialization of parameters count for counting the
    length of game, path records the step of the game to completion
    , token is actual element moving through the game
            count = 0
            path = []
            token = 0
            while token < 100:
                roll = random.randint(1, self.dice)
                token = token + roll
                count += 1
                for trans in self.sl:
                    if token == trans[0]:
                        token = trans[1]
                        break
                path.append(token)
            return [count, path]

    #simulation for the game with combination of 2 dice
    def gamesimulation4(self):
    # initialization of parameters count for counting the length of
     game, path records the step of the game to completion, token
    is actual element moving through the game
        count = 0
        path = []
        token = 0
        while token < 100:
            roll = random.randint(1, self.dice)
            roll2=random.randint(1,self.dice)
            #roll3=random.randint(1,self.dice)
            token = token + roll + roll2
            count += 1
            #snake or ladder transition
            for trans in self.sl:
                if token == trans[0]:
                    token = trans[1]
                    break
            path.append(token)
        return [count, path]

    #simulation for the game with combination of 3 dice
    def gamesimulation5(self):
    # initialization of parameters count for counting the length of
     game, path records the step of the game to completion, token
    is actual element moving through the game
        count = 0
        path = []
        token = 0
        while token < 100:
            roll = random.randint(1, self.dice)
            roll2=random.randint(1,self.dice)
            #roll3=random.randint(1,self.dice)
            token = token + roll + roll2
            count += 1
            # contorls token should land exactly at 101 or 100
            if token > 101:
```

```
                token = token - (roll + roll2)
            for trans in self.sl:
                if token == trans[0]:
                    token = trans[1]
                    break
            path.append(token)
        return [count, path]

    #simulation for harmonic distribution die
    def simulation_harmonic(self):
        count = 0
        path = []
        token = 0
        while token < 100:
            roll=random.choices(population=range(1,self.dice+1),k
    =1,weights=self.dist)[0]
            token+=roll
            count+=1
            if token>100:
                token-=roll
                count+=1
            for trans in self.sl:
                if token == trans[0]:
                    token = trans[1]
                    break
            path.append(token)
        return [count, path]
```

Listing 4: simulation for snakes and ladders game

```
#harmonic number algorithm
class Harmonic(object):
    def __init__(self,term=1):
        self.term = term

    def harmonic(self):
        h=1
        for i in range(2,self.term+1):
            h+=1/i
        return h
```

Listing 5: harmonic number algorithm

Declaration of Academic Integrity / Eidesstattliche Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind. Mit der aktuell geltenden Fassung der Satzung der Universität Passau zur Sicherung guter wissenschaftlicher Praxis und für den Umgang mit wissenschaftlichem Fehlverhalten vom 31. Juli 2008 (vABlUP Seite 283) bin ich vertraut. Ich erkläre mich einverstanden mit einer Überprüfung der Arbeit unter Zuhilfenahme von Dienstleistungen Dritter (z.B. Anti-Plagiatssoftware) zur Gewährleistung der einwandfreien Kennzeichnung übernommener Ausführungen ohne Verletzung geistigen Eigentums an einem von anderen geschaffenen urheberrechtlich geschützten Werk oder von anderen stammenden wesentlichen wissenschaftlichen Erkenntnissen, Hypothesen, Lehren oder Forschungsansätzen.

Passau, 04.08.2021

_____

Edwin Beatus Ismail

I hereby confirm that I have composed this scientific work independently without anybody else's assistance and utilising no sources or resources other than those specified. I certify that any content adopted literally or in substance has been properly identified. I have familiarised myself with the University of Passau's most recent Guidelines for Good Scientific Practice and Scientific Misconduct Ramifications from 31 July 2008 (vABlUP Seite 283). I declare my consent to the use of third-party services (e.g., anti-plagiarism software) for the examination of my work to verify the absence of impermissible representation of adopted content without adequate designation violating the intellectual property rights of others by claiming ownership of somebody else's work, scientific findings, hypotheses, teachings or research approaches.

Passau, 04.08.2021

_____

Edwin Beatus Ismail