Modeling Candy Land using a Markov Chain

Morgan Buterbaugh, Casey Sharek, Toby Trotta

December 2022

1 Introduction

Imagine you, a mathematician, sit down to play the game Candy Land with your friends. Everyone is focused on being the first to reach the Candy Castle, all while still having fun along the way. You, however, are thinking of so many questions. What is the average number of moves for one game, how long until someone reaches the Candy Castle, what is the least amount of moves to reach the Candy Castle, what is the probability of passing through the Rainbow Trail or the Gumdrop Pass, what are the chances that a player lands on a licorice space...? Then it hits you! You should model the game as a Markov Chain!

2 Markov Chains

To fully understand what a Markov chain really is, one should be familiar with stochastic processes.

Observe a system's characteristic at discrete points in time t, and we let X_t be the value of a specific system characteristic at time t. Generally, X_t is not known before time t. This is known as a stochastic process. A discrete-time stochastic process is a description of the relation between the random variables X_i for $i \in \{0, 1, 2, \dots\}$. Now with this information, we can define and explore Markov chains.

A Markov chain is a special type of discrete-time stochastic process. It is assumed that at any time t, our discrete-time stochastic process can be in any one of S distinct discrete states labeled 1, 2, ... , s. As defined in the Winston textbook: A discrete-time stochastic process is a Markov chain if, for t = 0, 1, 2, ... and all states, $P(X_{t+1} = i_{t+1} \mid X_t = i_t, X_{t-1} = i_{t-1}, \ldots, X_1 = i_1, X_0 = i_0) = P(X_{t+1} = i_{t+1} \mid X_t = i_t)$. Thus, the probability of the state at time $t = 1$ is dependent on the previous state at time t, and not the previous states the chain passed through to get to it.

For all states i and j and all t, $P(X_{t+1} = j \mid X_t = i)$ is independent of t. Then, $P(X_{t+1} = j \mid X_t = i) = p_{ij}$, where p_{ij} denotes the probability of a system that is in state j at time $t + 1$, given it was in state i at time t. Moving from state i to state j is called a transition, and the values of p_{ij} are known as the transition probabilities of the Markov Chain. [4]

When working with Markov chains, it is important to know the classifications of the states in a Markov chain. The two main classifications that we will work with are absorbing states and the mean first passage time.

An absorbing state is a state where $p_{ii} = 1$. Hence, an absorbing state is one in which once we enter, we never leave. Thus, a Markov chain that has at least one absorbing state is called an absorbing Markov chain. When setting up an absorbing Markov chain, the non-absorbing states are listed first, and then the absorbing states. Accordingly, the transition probability matrix is set up the same way, such that the absorbing states are the last rows and columns respectively. Once the transition matrix is established, there will be four distinct sections.

$$P = \left[\begin{array}{c|c} \text{N} & \text{A} \\ \hline 0 & \text{I} \end{array}\right]$$

The 'N' section contains all the non-absorbing states, the 'A' section contains the absorbing states, the '0' is a zero matrix, and the 'I' is an identity matrix. To calculate the probability of absorption in state j starting in state i, you would do the following calculation, $(I - N)^{-1}$.

The mean first passage time from state i to state j, or m_{ij}, is equal to the expected number of transitions before you reach state j, given that you started in state i. Hence, the mean first passage time is the expected number of transitions a chain must pass through before it reaches a given state. When setting up a probability transition matrix with a mean first passage time, the matrix is denoted N_j, where the j^{th} row and column are missing. To calculate the passage time into state j, the following calculation should be done, $m_j = (I - N_j)^{-1}\mathbf{1}$.

3 Candy Land Game & Rules

The rules discussed in this section were found from [3].

3.1 Overview:

Candy Land, published by Hasbro in 1949, is a game in which 2-4 players race to the Candy Castle at the top of the game board. Candy Land is a direction-based game where players must navigate through the board simply by drawing cards that include single or double color squares and pictures. The first to the Candy Castle is declared the winner.

3.2 Game Board:

The game board we will be using is provided in Figure 1. In this board, there are 134 spaces of which 22 are red, 21 are orange, 21 are yellow, 21 are green, 21 are blue, 21 are purple, 6 are pink (picture space), and one is rainbow, the final space. These spaces follow a general pattern: {red, purple, yellow, blue, orange, green}. The pink picture spaces are seemingly random and are not guaranteed to fall between the same two colors at each occurrence. For example, the Gingerbread is between a purple and a yellow space while the Candy Cane is between a green and red space.

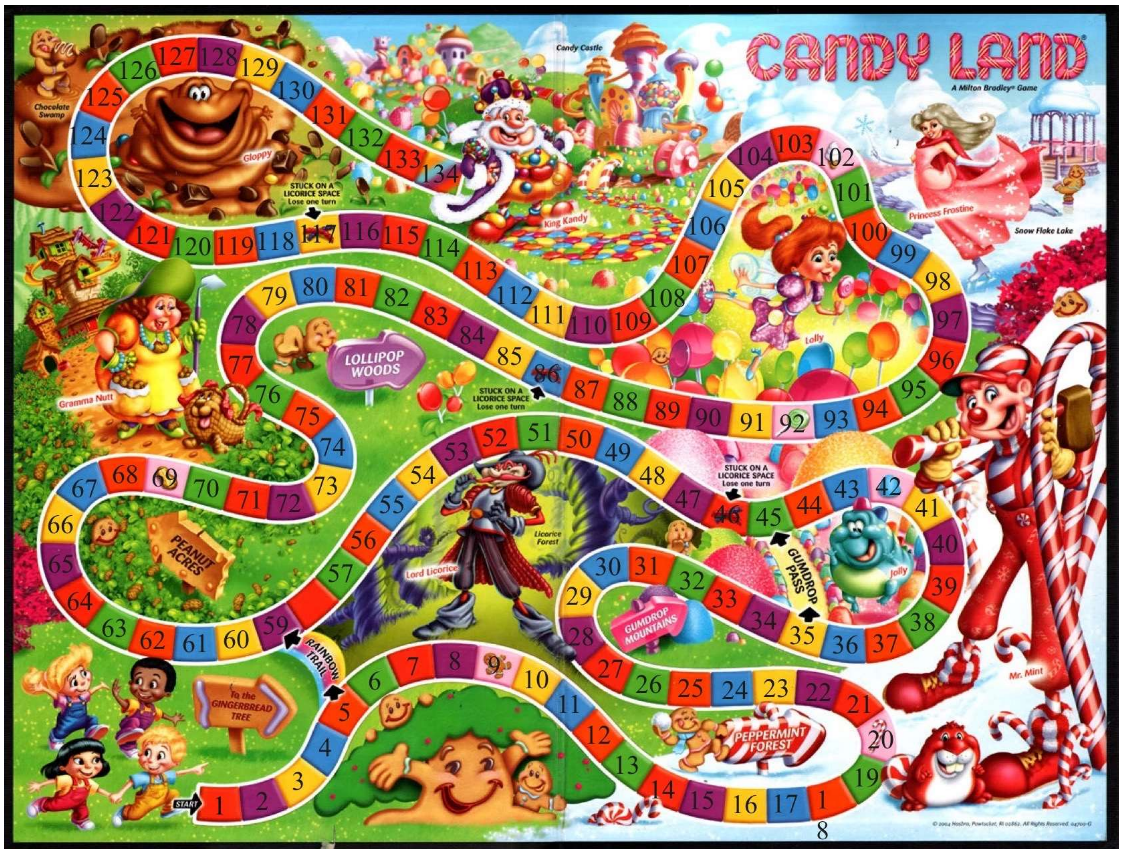

Figure 1: The Candy Land board that will be used.

3.3 Gameplay and Movement:

Starting at the bottom of the game board, players will take turns drawing a card. Without loss of generality, if a player draws a single red card, they will progress forward to the nearest red space. If they draw a double red card, they will move two red spaces forward. If a player draws a picture card containing one of the six items scattered throughout the board, the player must immediately move to that space. This may cause the player to skip a majority of the board or to move backward. There are no limitations to how many players may be in the same space.

Standard game rules rely on the players discarding their drawn card after moving their player piece. We will later discuss our decision to amend the rule to play *with* replacement rather than without. (This is important to how we model the game using a Markov Chain.)

Finally, there are three licorice spaces throughout the board. These are not picture cards included in the deck and only affect the game when landed on. If the player lands on a space with a licorice piece, the player loses a turn. Original game rules require the player to draw cards with certain conditions to leave the spot but those will be ignored for this analysis.

There was some difficulty in finding the official breakdown of the Candy Land deck considering newer versions utilize a spinner to move, however, based on similar analyses for Candy Land [2] and for purposes of this project, we will consider the following 64-card breakdown provided in Table 1.

Card Type	**Count**
Red (Single)	6
Red (Double)	4
Orange (Single)	6
Orange (Double)	3
Yellow (Single)	6
Yellow (Double)	4
Green (Single)	6
Green (Double)	3
Blue (Single)	6
Blue (Double)	4
Purple (Single)	6
Purple (Double)	4
Gingerbread Picture Card	1
Candy Cane Picture Card	1
Gumdrop Picture Card	1
Peanut Picture Card	1
Lollipop Picture Card	1
Ice Cream Picture Card	1

Table 1: Standard Candy Land card distribution.

3.4 Endgame

The first player to reach the Candy Castle is declared the winner. Toward the end of the board, a specific color is not needed in order for a player to win. For example, if the player is on spot 131 and they choose any double color card or a single color card for the colors orange, blue, yellow, or purple, choosing any of these cards will result in a win. The same rules apply if a special card is drawn; the player must be taken back to that specific picture card.

4 Modeling Candy Land as a Markov Chain

Before the transition matrix can be created, a few adjustments must be made to the Candy Land rules. First of all, after a card is drawn in a typical game of Candy Land, it must be placed into a discard pile, and once the entire deck has been used it can be shuffled and used again. However, because the game is being modeled using a Markov Chain, this rule must be changed. Rather than discarding the card, it needs to be replaced in the deck, and the deck must then be shuffled after every player's turn. This allows the chances of choosing any card to be constant and completely random.

It is not possible for a game to be modeled using a Markov Chain without the replacement and shuffling adjustments. Not only would the calculations for each transition be far too complex and convoluted, but they would be impossible. The Markov Chain is only modeling the movements of one player, so there is no way to know what the other players in the game would have chosen on their own turns, thus affecting the calculations for the player's next turn. Additionally, the chain does not take into account how many people are playing the game.

For a simple example, look at the situation where a player chooses a single green card on their first turn. There are multiple factors that affect the chances that the player chose that single green card. Was this player going first, second, third, or fourth? And if they weren't the first player to go, what card did the player(s) before them choose? Table 2 outlines the possible situations the players could have found themselves in. Because there are external factors that affect the chances of a player choosing a card when replacement is not being used, this rule is necessary to use a Markov Chain to model Candy Land.

When does the player go?	Chances of Choosing Green
First	$\frac{6}{64}$
Second	$\frac{6}{63}$ or $\frac{5}{63}$
Third	$\frac{6}{62}$ or $\frac{5}{62}$ or $\frac{4}{62}$
Fourth	$\frac{6}{61}$ or $\frac{5}{61}$ or $\frac{4}{61}$ or $\frac{3}{61}$

Table 2: Each of the different situations the player can find themselves in depends on the previous players' moves.

4.1 Creating the Transition Matrix

In order to create the transition matrix of a Candy Land board, each position on the board needs its own state in the matrix. This includes the beginning spot, which is not on any space on the board. In the transition matrix, this spot will be the first row and column, with an index of 0. Each spot will be in order from $1-134$ with 134 being the final spot on the board.

On spots 5 and 35, there are shortcuts. So that the indexing in the transition matrix still lines up with the spot numbers, the Rainbow Trail and the Gumdrop Pass shortcuts will have two states in the transition matrix. The Rainbow Trail takes the player from position 5 to 59. Both rows 5 and 59 in the transition matrix will be the same. The Gumdrop Pass will take the player from position 35 to 45, and both rows 35 and 45 will have that same transition states probabilities.

For the licorice spaces, an additional state is added for each licorice space. A similar process is used in [1] in using a Markov Process to model the jail spaces in a game of Monopoly. These licorice states correspond to index 135, 136, and 137. These additional states are needed in order to simulate a player losing a turn. For example, when the player lands on space 46 (licorice space #1), on their next turn, they have a 100% chance to transition from state 46 to state 135, simulating a lost turn. Then, on their following turn, they would have the same chances to move from state 135 to another state as if there was no licorice piece on space 46. To clarify, the player has a $\frac{6}{64}$ chance to move from state 135 to state 47, a $\frac{6}{64}$ chance to move from state 135 to state 48, and so on.

Each picture card has the same chance every round for a player to land on it. For this reason, the column of each picture card has the same value for every element. With the assumptions and special cases outlined, the transition matrix can be constructed as shown below. To make the matrix more legible, each element in the matrix represents the number of cards that can take a player to that position, and the matrix is then multiplied by $\frac{1}{64}$ to get the probability transition matrix.

$$T = \frac{1}{64}\begin{bmatrix}
0 & 6 & 6 & 6 & 6 & 6 & 6 & 4 & 4 & 1 & 4 & 4 & 3 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 6 & 6 & 6 & 6 & 6 & 6 & 4 & 1 & 4 & 4 & 3 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 6 & 6 & 6 & 6 & 6 & 6 & 1 & 4 & 4 & 3 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 6 & 6 & 6 & 6 & 6 & 1 & 6 & 4 & 3 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 6 & 6 & 6 & 6 & 1 & 6 & 6 & 3 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & 1 & 6 & 6 & 6 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & & & & & & & & & & & & & \ddots & & & & & & & & & & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 6 & 6 & 6 & 40 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 6 & 6 & 46 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 6 & 52 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 58 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 64 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

The code for creating the transition matrix can be found in the appendix. Note that rather than typing out the entire matrix, a general pattern was found for the transition matrix, and from there adjustments were made to accurately reflect the transition matrix.

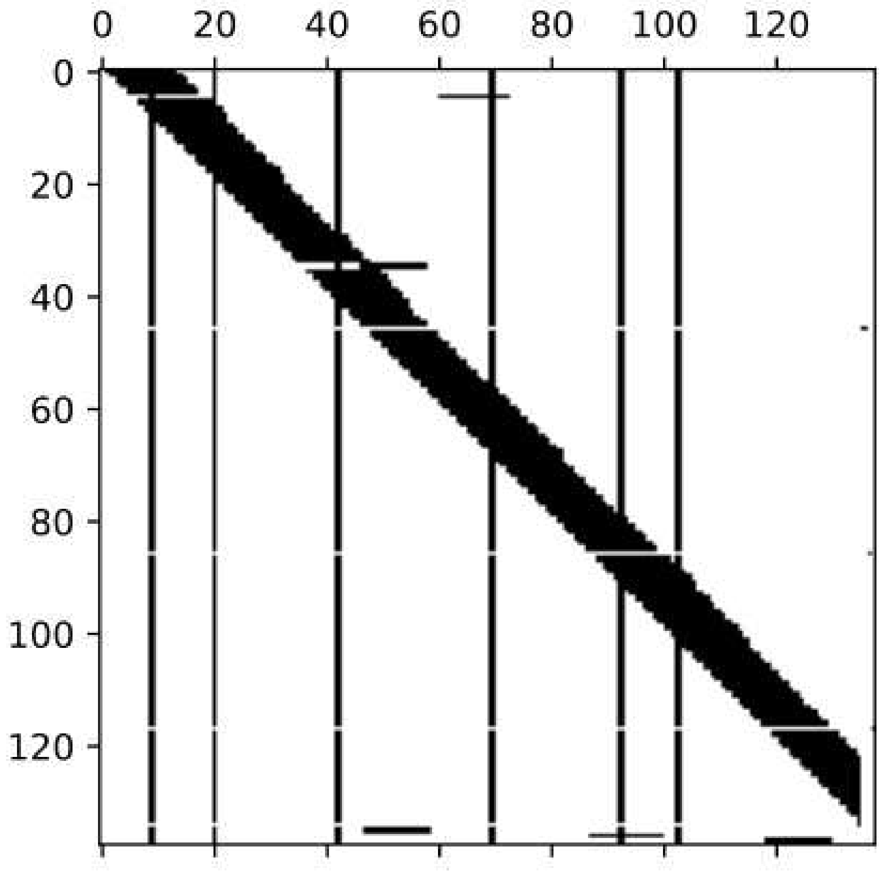

Figure 2: A spy plot of the transition matrix for Candy Land. Black spots are non-zero values, and white spots are zeros.

This was a more effective way of creating the transition matrix since the majority of the elements in the matrix are zeros. To get an understanding of the amount of zeros in the matrix, Figure 2 is a spy plot of the transition matrix.

4.2 Average Number of Moves in a Game of Candy Land

In order to find the amount of turns a typical game of Candy Land will last, separating the matrix into its absorbing and non-absorbing states will help accomplish this. To do this, the absorbing states must be identified; the absorbing states are the states where their rows in the transition matrix look like a row from an identity matrix. In the case of this Candy Land board rows 46, 86, 117, and 134 are considered the absorbing states. These rows correspond to licorice piece 1, licorice piece 2, licorice piece 3, and the final spot on the board. These rows and their corresponding columns must be deleted from the original transition matrix P. The fundamental matrix, which gives the expected time in state j starting in state i, can be found using the following equation:

$$(I - N)^{-1}$$

where N is the 134x134 P matrix without rows 46, 86, 117, and 134 or columns 134, 135, 136, and 137. Once the fundamental matrix is known, finding the vector

$$(I - N)^{-1}\mathbf{1}$$

will result in a vector of values that represent the expected number of turns to absorption starting in state i. The resulting vector is:

$$(I - N)^{-1}\mathbf{1} = \begin{bmatrix} 27.87966595 \\ 27.86937312 \\ 27.85249506 \\ 27.82922326 \\ 27.80008721 \\ 22.73524559 \\ \vdots \\ 8.14432931 \\ 5.85154385 \\ 4.14503883 \\ 3.07705552 \end{bmatrix}$$

When looking at the vector, the first value is the number of expected moves it will take the player to reach the absorbing state if they started at the beginning of the board game.

So, starting at the beginning of the board, the expected amount of turns that a game of Candy Land will last is 27.8797 moves, which can be rounded to roughly 28 moves per player.

Notice how the number of moves tends to decrease the higher the index gets with the exception of a few positions on the board. This is because the farther into the board a player gets, the less amount of moves they're expected to make to get to the end of the board.

4.2.1 Expected Time to Reach the Candy Castle

For purposes of this project, let's assume that each turn takes 8 seconds from drawing a card, moving your piece to the appropriate state, and discarding the drawn card. If we expect, by the previous section, that a game lasts approximately 28 moves, we can expect it takes $28 \cdot \left(\frac{8}{60}\right) \approx 3.75$ minutes for a single person. Since a standard Candy Land box suggests 2-4 players per game, we can expect a full 4-person game to last anywhere between 15-20 minutes.

4.3 Least Amount of Moves Possible

This question is easy to analyze without the transition matrix. Starting at the beginning of the game, the card that takes the player the furthest in one move is the ice cream cone card. This card takes the player to position 102 on the board. From there, the player must choose some combination of three different double-color cards. By doing this, the player can win the game in only four moves. However, the likelihood that a player can pull this off is slim. The final three cards can be a variety of double-color cards; there are ten different combinations of cards that can result in a win. The following table is the different card combinations that can result in a player winning in just four cards. The cards must be drawn in the order presented in the table in order for the combinations to work.

First Move	Second Move	Third Move	Fourth Move
Ice Cream Cone ($\frac{1}{64}$)	Double Green ($\frac{3}{64}$)	Double Orange ($\frac{3}{64}$)	Double Blue ($\frac{4}{64}$)
Ice Cream Cone ($\frac{1}{64}$)	Double Green ($\frac{3}{64}$)	Double Orange ($\frac{3}{64}$)	Double Yellow ($\frac{4}{64}$)
Ice Cream Cone ($\frac{1}{64}$)	Double Green ($\frac{3}{64}$)	Double Orange ($\frac{3}{64}$)	Double Purple ($\frac{4}{64}$)
Ice Cream Cone ($\frac{1}{64}$)	Double Green ($\frac{3}{64}$)	Double Blue ($\frac{4}{64}$)	Double Yellow ($\frac{4}{64}$)
Ice Cream Cone ($\frac{1}{64}$)	Double Green ($\frac{3}{64}$)	Double Blue ($\frac{4}{64}$)	Double Purple ($\frac{4}{64}$)
Ice Cream Cone ($\frac{1}{64}$)	Double Green ($\frac{3}{64}$)	Double Yellow ($\frac{4}{64}$)	Double Purple ($\frac{4}{64}$)
Ice Cream Cone ($\frac{1}{64}$)	Double Orange ($\frac{3}{64}$)	Double Blue ($\frac{4}{64}$)	Double Yellow ($\frac{4}{64}$)
Ice Cream Cone ($\frac{1}{64}$)	Double Orange ($\frac{3}{64}$)	Double Blue ($\frac{4}{64}$)	Double Purple ($\frac{4}{64}$)
Ice Cream Cone ($\frac{1}{64}$)	Double Orange ($\frac{3}{64}$)	Double Yellow ($\frac{4}{64}$)	Double Purple ($\frac{4}{64}$)
Ice Cream Cone ($\frac{1}{64}$)	Double Blue ($\frac{4}{64}$)	Double Yellow ($\frac{4}{64}$)	Double Purple ($\frac{4}{64}$)

Since the player can win by drawing any of the card combinations above, the calculations to find the chances of this occurring is relatively simple. The chances of each card being drawn is provided in the table, and the following calculation will find the chance of a player winning in only four moves:

$$\begin{aligned} P(\text{Win in Four Moves}) &= 3\left(\frac{1}{64}\right)\left(\frac{3}{64}\right)^2\left(\frac{4}{64}\right) + 6\left(\frac{1}{64}\right)\left(\frac{3}{64}\right)\left(\frac{4}{64}\right)^2 + \left(\frac{1}{64}\right)\left(\frac{4}{64}\right)^3 \\ &= \frac{115}{4194304} \approx 0.0000274 \end{aligned}$$

So, the least amount of moves a player can make to win the game is four, but they have a very low chance of doing so, at only 0.00274% of the time winning in only four moves.

4.4 Reaching Special States

4.4.1 What is the probability a player may pass through the Rainbow Trail or the Gumdrop Pass?

To determine the probability of taking either shortcut, we must amend our original transition matrix. By letting states 5 and 35, the entrances to the Rainbow Trail and Gumdrop Pass, respectively, be absorbing states, we can determine the probability of landing on each space.

The spy plots below represent the non-absorbing and absorbing matrices used for this section and the following section where we mention the licorice spaces, Figures 3 and 4:

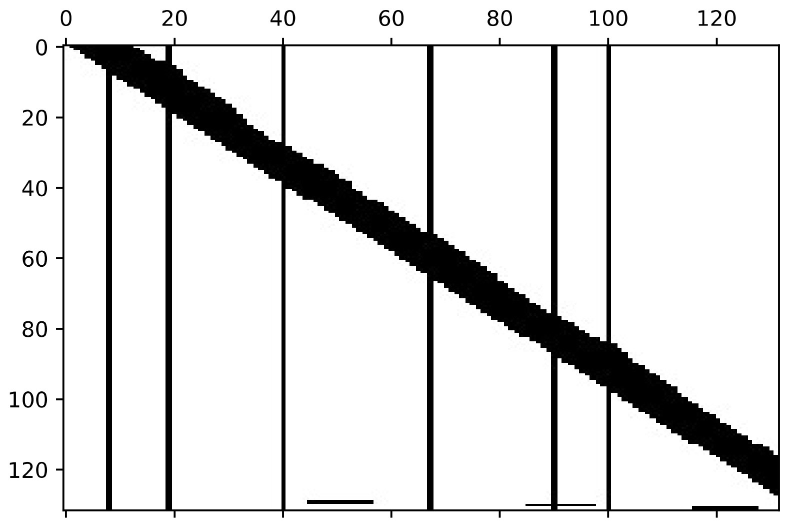

Figure 3: A spy plot of the non-absorbing matrix, N.

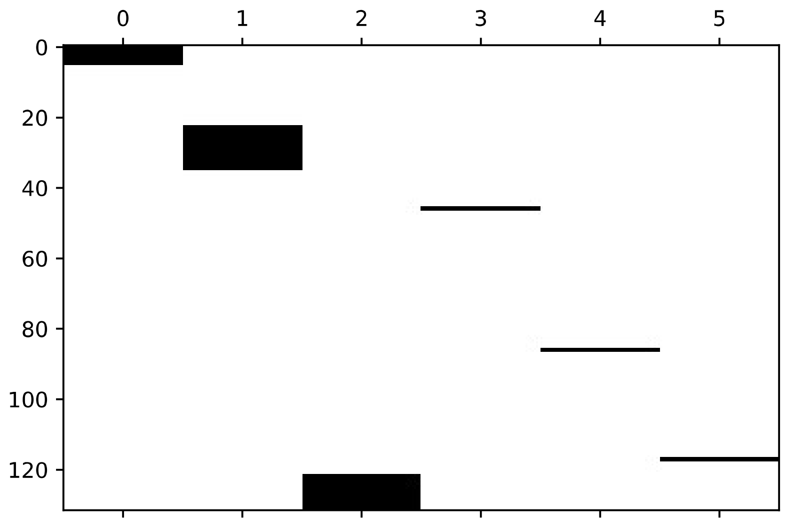

Figure 4: A spy plot of the absorbing matrix, A.

After the adjustments are made to find the probability of absorption then splitting our transition matrix into the appropriate non-absorbing and absorbing matrices the dot-product of $(I - N)^{-1}$ and A result in the following truncated matrix:

$$\begin{bmatrix} 0.26833 & 0.28747 \\ 0.24533 & 0.29392 \\ 0.22430 & 0.29999 \\ 0.20508 & 0.30554 \\ 0.18750 & 0.31086 \\ \vdots & \vdots \end{bmatrix}$$

Since there are no picture cards that bring the player before the entrance to the Rainbow Trail, there is no probability of taking it once you progress past the entrance. However, there are two picture cards (gingerbread and candy cane) that are before the Gumdrop Pass, so a player may find themselves taking the Gumdrop Pass must later in the game.

4.4.2 What are the chances a player lands on a licorice space?

By our original transition matrix, any of the three licorice spaces on the board can be considered absorbing states so no amendments must be made.

After a similar process of splitting our transition matrix into non-absorbing and absorbing states is complete, we are given the truncated matrix. The probabilities of landing on either licorice space 1 (state 46), licorice space 2 (state 86), or licorice space 3 (state 117), in order from left to right are:

$$\begin{bmatrix} 0.15077 & 0.11170 & 0.13210 \\ 0.15334 & 0.11325 & 0.13359 \\ 0.15577 & 0.11470 & 0.13495 \\ 0.15803 & 0.11601 & 0.13615 \\ 0.16015 & 0.11722 & 0.13723 \\ 0.177783 & 0.12950 & 0.15095 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

We saw that, as you continue down the matrix, as expected, the probability of landing on licorice space 1 (state 46) significantly decreases. Furthermore, we noticed that the probability of landing on licorice space 3 (state 117) increases as the game progresses. This is because you are progressing toward the licorice space, thus a higher likelihood of landing on the spot.

Contrary to the results of reaching the Rainbow Pass in the previous section, the player always has some probability of landing on any of the three licorice spaces at any time throughout the game thanks to the picture cards scattered throughout.

4.5 Conclusions:

Modeling Candy Land as a Markov Chain allows us to find different probabilistic characteristics of the game. By making a transition matrix for a single player, we were able to find the average amount of moves a player will make in a game of Candy Land, and as a result the expected time a game will take. In addition to these expected values, we were able to find the chances of a player landing on the special spots on the board: licorice, the Rainbow Trail, and Gumdrop Pass. In further research, chances of landing on the pictures will be analyzed.

References

[1] R. Ash and R. Bishop. Monoploy as a markov process. 1972.

[2] N. Berry. Mathematical analysis of candy land, 2011. Accessed: November 20, 2022.

[3] G. Carreau. How to play candy land: Easy gameplay guide and rules, 2022. Accessed: November 17, 2022.

[4] Wayne L. Winston. *Operations Research Applications and Algorithms.* Thomson Brooks/Cole, fourth edition, 2003. Accessed: October 15, 2022.

Appendix

Listing 1: Python code for the construction of the transition matrix.

```
import numpy as np
from math import *
import sys
np.set_printoptions(threshold=sys.maxsize)

T = np.zeros(138*138)
T = T.reshape(138,138)

# The general pattern for the matrix
# Starting at position 0, which is the starting point
for i in range(134):
    T[i,i+1]=6
for i in range(133):
    T[i,i+2]=6
for i in range(132):
    T[i,i+3]=6
for i in range(131):
    T[i,i+4]=6
for i in range(130):
    T[i,i+5]=6
for i in range(129):
    T[i,i+6]=6
for i in range(128):
    T[i,i+7]=4
for i in range(127):
    T[i, i+8]=4
for i in range(126):
    T[i, i+9]=4
for i in range(125):
    T[i, i+10]=4
for i in range(124):
    T[i, i+11]=4
for i in range(123):
    T[i, i+12]=4

# Fixing the 3's
for i in range(0,5):
    T[i,12]=3
    T[i,13]=3
for i in range(6,12):
    T[i,18]=3
    T[i,19]=3
T[12,19]=3
for i in range(12,18):
    T[i,25]=3
    T[i+1,26]=3
for i in range(18,25):
    T[i,31]=3
    T[i+1,32]=3
for i in range(25,31):
    T[i,37]=3
    T[i+1,38]=3
```

```
for i in range(31,37):
    T[i,44]=3
    T[i+1,45]=3
for i in range(37,44):
    T[i,50]=3
    T[i+1,51]=3
for i in range(44,50):
    T[i,56]=3
    T[i+1,57]=3
for i in range(50,56):
    T[i,62]=3
    T[i+1,63]=3
for i in range(56,62):
    T[i,68]=3
    T[i+1,70]=3
for i in range(62,68):
    T[i,75]=3
    T[i+1,76]=3
T[68,81]=3
T[69,76]=3
for i in range(69,75):
    T[i,81]=3
    T[i+1,82]=3
for i in range(75,81):
    T[i,87]=3
    T[i+1,88]=3
for i in range(81,87):
        T[i,94]=3
        T[i+1,95]=3
    for i in range(87,94):
        T[i,100]=3
        T[i+1,101]=3
    for i in range(94,100):
        T[i,107]=3
        T[i+1,108]=3
    for i in range(100,107):
        T[i,113]=3
        T[i+1,114]=3
    for i in range(107,113):
        T[i,119]=3
        T[i+1,120]=3
    for i in range(113,119):
        T[i,125]=3
        T[i+1,126]=3
    for i in range(119,125):
        T[i,131]=3
        T[i+1,132]=3

    # Fixing row 5, which will be for position #59 (the rainbow trail)
    for i in range(len(T)):
        T[5,i]=0
    for i in range(60,67):
        T[5,i]=6
    T[5,66],T[5,67]=4,4
```

```
T[5,68],T[5,70]=3,3
T[5,71],T[5,72]=4,4

# Fixing row 35, which will be for position #45 (gumdrop pass)
for i in range(len(T)):
    T[35,i]=0
for i in range(46,52):
    T[35,i]=6
for i in range(52,56):
    T[35,i]=4
T[35,56],T[35,57]=3,3

# For the picture cards
for i in range(0,len(T)):
    T[i,9] = 1
for i in range(0,len(T)):
    T[i,20] = 1
for i in range(0,len(T)):
    T[i,42] = 1
for i in range(0,len(T)):
    T[i,69] = 1
for i in range(0,len(T)):
    T[i,92] = 1
for i in range(0,len(T)):
    T[i,102] = 1

# Row 135 for licorice #1
```

```
for i in range(47,53):
    T[135,i]=6
for i in range(53,56):
    T[135,i]=4
T[135,56],T[135,57]=3,3
T[135,58]=4
for i in range(len(T)):
    T[46,i]=0
T[46,135]=64

# Row 136 for licorice #2
for i in range(87,92):
    T[136,i]=6
for i in range(96,100):
    T[136,i]=4
T[136,93]=6
T[136,94]=3
T[136,95]=3
for i in range(len(T)):
    T[86,i]=0
T[86,136]=64

# Row 137 for licorice #3
for i in range(118,124):
    T[137,i]=6
for i in range(124,130):
    T[137,i]=4
```

```
T[137,125]=3
T[137,126]=3
for i in range(len(T)):
    T[117,i]=0
T[117,137]=64

#adjustments for the picture card columns
for i in range(0,4):
    T[1+i,14+i]=4
    T[7,21]=4
T[8,21]=4
T[8,22]=4
for i in range(0,3):
    T[9+i,22+i]=4
T[3,10]=6
T[4,11]=6
for i in range(0,3):
    T[6+i,13+i]=6
for i in range(0,4):
    T[14+i,27+i]=4
for i in range(0,6):
    T[14+i,21+i]=6
T[30,43]=4
for i in range(0,4):
    T[33+i,46+i]=4
for i in range(0,6):
    T[36+i,43+i]=6
```

```
for i in range(0,3):
    T[39+i,52+i]=4
for i in range(0,4):
    T[58+i,71+i]=4
for i in range(0,6):
    T[63+i,70+i]=6
for i in range(0,4):
    T[64+i,77+i]=4
T[80,93]=4
for i in range(0,3):
    T[83+i,96+i]=4
for i in range(0,3):
    T[87+i,94+i]=6
T[89,103]=4
for i in range(0,6):
    T[96+i,103+i] = 6
for i in range(0,4):
    T[96+i,109+i]=4
for i in range(0,2):
    T[90+i,97+i]=6
    T[90,103+i]=4
    T[91,104+i]=4
T[92,105]=4
T[93,106]=4
T[35,48]=6

# Row 134 should be all 0's except 134 since it's the end of the game
```

```
for i in range(len(T)):
    T[134,i]=0
T[134,134]=64

# Column 134 is the winning state, so the numbers need to cumulate there
T[123,134]=8
T[124,134]=12
T[125,134]=15
T[126,134]=18
T[127,134]=22
T[128,134]=28
T[129,134]=34
T[130,134]=40
T[131,134]=46
T[132,134]=52
T[133,134]=58

print(T)
#Trow =[]
#for i in range(len(T)):
 #    Trow.append(T[112,i])
#print(Trow)

T = (1/64)*T
print(T)
```