math381HW5

Siyue Zhu

February 2022

1 Introduction

This report is going to investigate a game using Markov Chain. The Markov Chains are very helpful when we solving question about modeling certain random processes that change discretely over a set of states. We can use a matrix A to show our Markov Chain, with rolls standing for the current state, and column representing the next state. And each entry A_{ij} tells us the probability of getting to state j after our current state i. Also, A^k can be calculated by A to the power of k. The entries of A^k_{ij} tells us the probability that we will have state j k steps after current state i.

Here is the rule of the game: we start at zero, and we have a goal of M. On each turn, we roll a six-sides die. The game will be ended once the sum of current score plus the new roll is equal or larger to M. If my current score plus the new roll is a prime number, then we use our current score minus the new roll. Any negative number would be considered as zero. On the other hand, if our current score plus the new roll is not a prime, then we use the current score to plus the new roll.

2 The Expected Value

In this question, we have M = 15. So our states are 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15, which means we have 15 different state before we can end the game.Now, let's discuss the expected number of the number of rolls made before the game ends.

First, we need to construct a transition matrix. A transition matrix is a matrix with row number representing the current state, and column number representing the next state. Let's say we have a transition matrix P, then ij-th entry is the probability that the chain will be in state j after starting in state i.

I constructed a transition matrix in sage, and the code can be found in the appendix. Here is the matrix I get:

$$A = \frac{1}{6}\begin{pmatrix}
3 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6
\end{pmatrix}$$

Now, we have the theorem:

$$P = \begin{pmatrix} Q & R \\ O & J \end{pmatrix}$$

J is the identity matrix, and O is a matrix for all zeros. Q and R are non-negative matrices that arise from the transition probabilities between non-absorbing states. We know the series $N = I + Q + Q^2 + Q^3 + ...$ converges, and $N = (I - Q)^{-1}$, then we can get matrix N. And the code for calculation matrix N can be found in the appendix.

Matrix N can give us a lot of information:

1. The ij-th entry of N is the expected number of steps that the chain will be in state j after starting in state i.
2. The sum of the i-th row of N gives the mean number of steps of starting at i, then until absorption.
3. The ij-th entry of the matrix B = NR is the probability after starting in non-absorbing state i, then end up in absorbing state j.

By summing the first row of matrix N, we get the value of the expected value of the number of steps until we end the game.

$$\sum_{i=0}^{15} iP_i = \frac{44248376}{4643277} = 9.52955768092233$$

3 Simulation

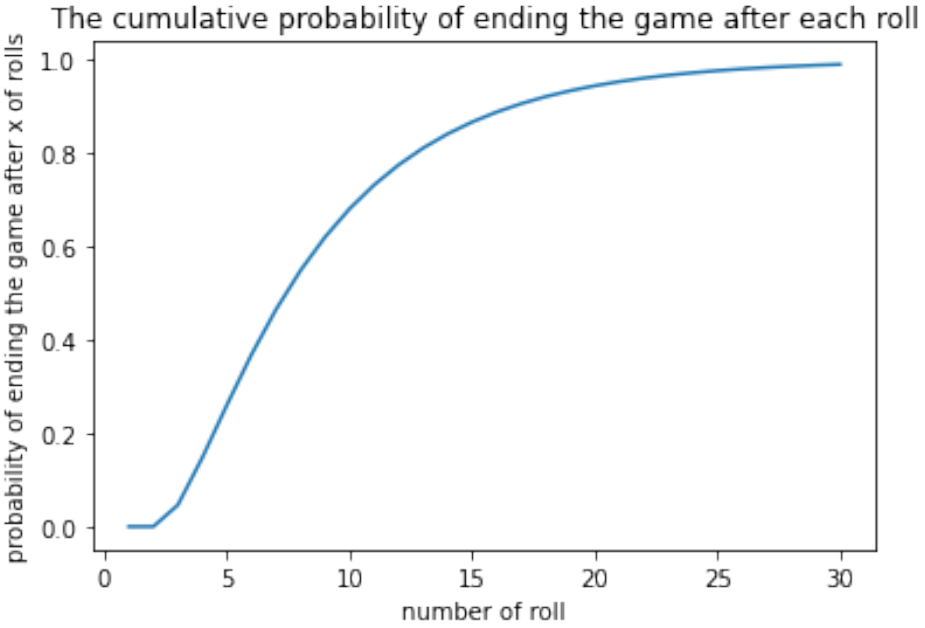
Now, I'm going to write a simulation to support my expected value in the previous part. I will run the simulation for 10 times with 10000 simulations games each. The code of running the simulation can be found in the appendix. Here is the list of 10 simulation result I get: 9.4297, 9.4598, 9.5288, 9.5334, 9.5375, 9.5505, 9.5514, 9.566, 9.5956, 9.6217. These results are all very close to our expected value from the previous part, thus the expected we calculated in the previous part is reliable. We can see the rage is from 9.4297 to 9.6217, and our expected value is in this range.

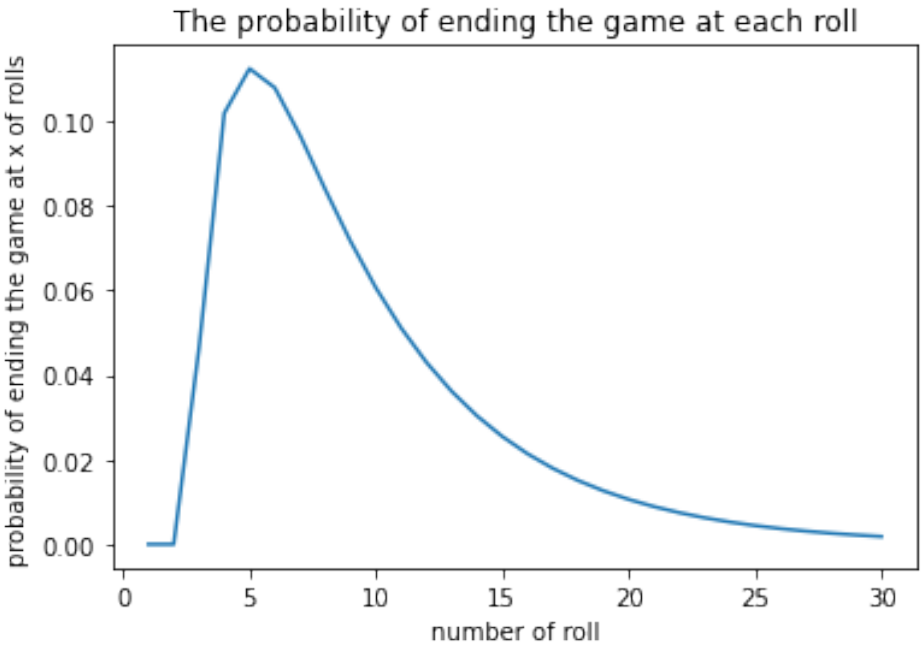
4 Investigate the Distribution

Now, let's investigate the distribution of the number of rolls we need to end the game.

Probability to end the game for each roll		
i	prob of ending game after i-th roll	prob of ending game on i-th roll
1	0	0
2	0	0
3	0.0462962962962963	0.0462962962962963
4	0.148148148148148	0.101851851851852
5	0.260545267489712	0.112397119341564
6	0.368419924554184	0.107874657064472
7	0.465009859396433	0.0965899348422496
8	0.548792700236244	0.0837828408398110
9	0.620334747148555	0.0715420469123101
10	0.680923149497663	0.0605884023491083
11	0.732017447793182	0.0510942982955187
12	0.775007918384725	0.0429904705915438
13	0.811136656445807	0.0361287380610815
14	0.841479538098792	0.0303428816529847
15	0.866954505505081	0.0254749674062889
16	0.888338656613678	0.0213841511085973
17	0.906287173428050	0.0179485168143724
18	0.921351260630962,	0.0150640872029122
19	0.933994116415223	0.0126428557842603
20	0.944604747266997	0.0106106308517742
21	0.953509745649391	0.00890499838239411
22	0.960983256691959	0.00747351104256844
23	0.967255379513996	0.00627212282203615
24	0.972519234480676	0.00526385496668003
25	0.976936901742375	0.00441766726169990
26	0.980644408427229	0.00370750668485320
27	0.983755915596161	0.00311150716893260
28	0.986367232785816	0.00261131718965479
29	0.988558767867415	0.00219153508159942
30	0.990398002833709	0.00183923496629401

Here are two graphs showing the distribution. The first graph is the probability of ending the game after each roll, and the second graph is the probability of ending the game at each roll.





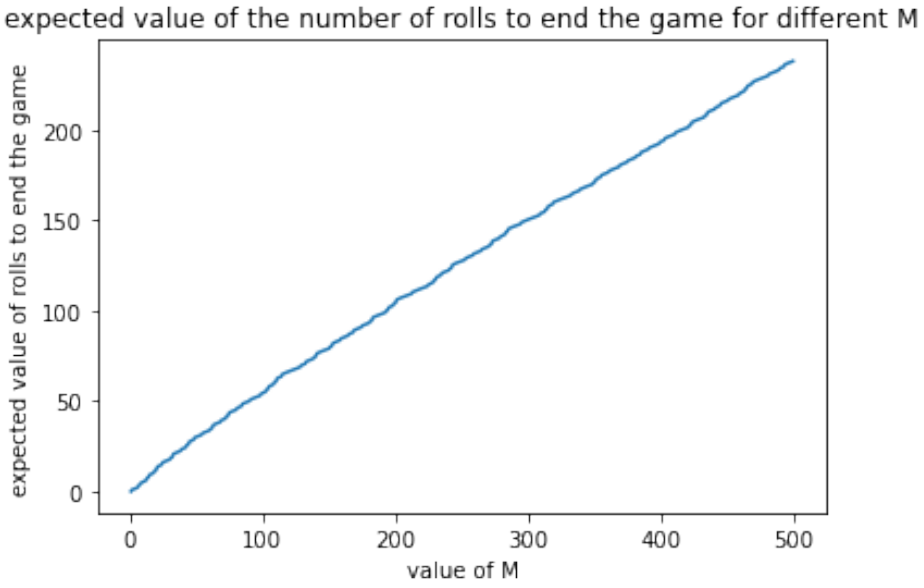
We can see there is 0 probability to end the game in the first two rolls. And the probability get larger until 5th roll. It is obvious to see there is a peak at x=5 in the second graph, which means the probability of ending the game at 5th roll is larger than other rolls. And after the 5th roll, the probability goes down and get close to 0 around 30th roll.

According to the graph, 46.5009859396433% of the game would end after the 7th roll, and 54.8792700236244% of the game would end after the 8th roll.So the median game length is between 7 and 8. And theoretically, we can end 95.3509745649391% of the game after the 21st roll, and we can end 99.0398002833709% of the game after the 30th roll.

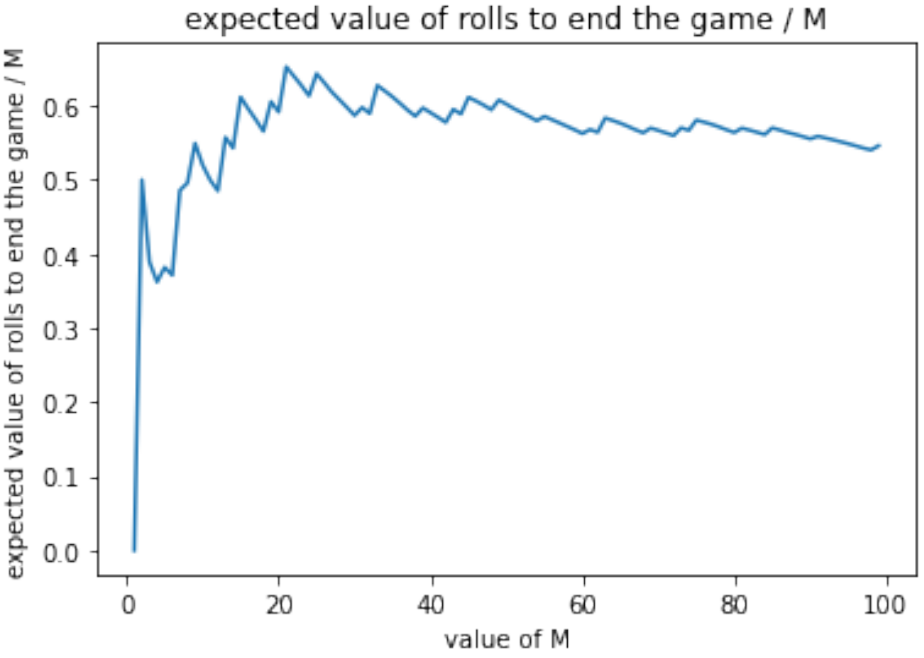
5 Investigate the Expected Number of Rolls for Different M

So far, we discussed the game under M = 15. Now, let's change the value of M and see how is the expected number of rolls change over different M.

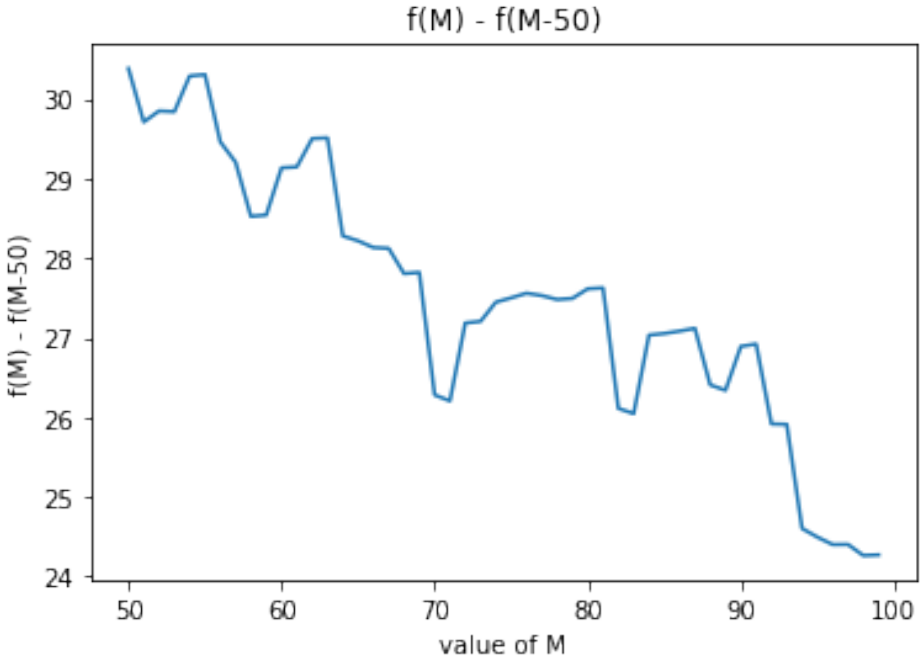
I generate code to calculate the expected number of rolls to end the game from M = 1 to M = 100. The code can be found in the appendix. And here is the graph showing how the expected value change for different value of M.



Since there is no solid bound of number of rolls to end the game, then M can continue getting larger and larger. I calculated the expected value of the number of rolls to end the game for M from 1 to 500, as the graph shows above. We can see as M getting larger, the expected value is also getting larger. And I stop at M = 100, since it takes a very long time for my laptop to solve the expect value for large M. Thus, I think I can calculate the expected value for very larger M with no limit, however the only constraint is the CPU and memory of my laptop. When I'm running my code on Jupyter Notebook online, the CPU turns in red when M gets larger. Thus, I think the CPU is not big enough for me to calculate the expected value for large M. Also, we can see the graph is not linear, but is very similar to a linear graph. And there is no limiting behavior.



The above graph is f(M)/M. If f(M) is linear as f(M) = kM, which k is a constant. Then f(M)/M should be equal to k, which is a vertical straight line on the graph. However, our graph is not a vertical straight line. Thus, f(M) is not linear. The irregular pattern of the graph is caused by the situation that we encounter prime in the game, so the expected value of rolls to end the game / M is not regular.



The above graph indicates the difference between f(M) and f(M-50) from m = 50 to M = 100. And we can see the graph is in a concave down shape. f(M) - f(M-50) represents the different of expected value for M with a difference of 50. According to the shape of the above graph, we can conclude f(M) is not a linear function.

6 appendix

```
A=zero_matrix(QQ,16);
for i in range(0,16):
    for j in range(1,7):
            if i+j >= 15:
                A[i,15] = A[i,15] + 1/6
            elif i+j not in Primes():
                A[i,i+j] = A[i,i+j] + 1/6
            elif i-j <= 0:
                A[i,0] = A[i,0] + 1/6
            else:
                A[i,i-j] = A[i,i-j] + 1/6

Q = A[:15,:15]
N=(matrix.identity(15)-Q).inverse()

sum = 0
for i in range(15):
    sum += N[0][i]
sum

sum = 0
for i in range(15):
    sum += N[0][i]
sum*1.

import random
```

```
for j in range(0,10):
    total = 0
    for i in range(0,100000):
        sum = 0
        step = 0
        while sum < 15:
            step = step + 1
            a = random.sample(range(1,7), 1)
            if sum + a[0] >=15:
                break
            elif sum + a[0] not in Primes():
                sum = sum + a[0]
            elif sum - a[0] <= 0:
                sum = 0
            else:
                sum = sum - a[0]
        total = total + step
    print(total/10000*1.)

list1 = []
for i in range(1, 31):
    list1.append((A^i)[0,15]*1.)
list1

list2 = [A[0,15] - 0]
for i in range(2,31):
    list2.append((A^i)[0,15]*1. - (A^(i-1))[0,15])
list2

import matplotlib.pyplot as plt
x = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30]
plt.plot(x,list1)
plt.xlabel('number of roll')
plt.ylabel('probability of ending the game after x of rolls')
plt.title('The cumulative probability of ending the game after each roll')

import matplotlib.pyplot as plt
x = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30]
plt.plot(x,list2)
plt.xlabel('number of roll')
plt.ylabel('probability of ending the game at x of rolls')
plt.title('The probability of ending the game at each roll')

list = []
for k in range(1,100):
    A=zero_matrix(QQ,k);
    for i in range(0,k):
        for j in range(1,7):
            if i+j >= k-1:
                A[i,k-1] = A[i,k-1] + 1/6
            elif i+j not in Primes():
                A[i,i+j] = A[i,i+j] + 1/6
            elif i-j <= 0:
                A[i,0] = A[i,0] + 1/6
            else:
                A[i,i-j] = A[i,i-j] + 1/6
```

```
    Q = A[:k-1,:k-1]
    N=(matrix.identity(k-1)-Q).inverse()
    sum = 0
    for i in range(k-1):
        sum += N[0][i]
    print(k)
    list.append(sum*1.)
list

import matplotlib.pyplot as plt
import numpy as np
x = []
for i in range(1,100):
    x.append(i)
x = np.array(x)
plt.plot(x,list)
plt.xlabel('value of M')
plt.ylabel('expected value of rolls to end the game')
plt.title('expected value of the number of rolls to end the game for different M')

x = []
for i in range(1,100):
    x.append(i)
x = np.array(x)
plt.plot(x,list/x)
plt.xlabel('value of M')
plt.ylabel('expected value of rolls to end the game / M')
plt.title('expected value of the number of rolls to end the game for different M')

x = []
for i in range(50,100):
    x.append(i)
x = np.array(x)
list_new = []
for i in range(50,100):
    now = list[i] - list[i-50]
    list_new.append(now)
plt.plot(x,list_new)
plt.xlabel('value of M')
plt.ylabel('f(M) - f(M-50)')
plt.title('f(M) - f(M-50)')
```