Assignment 5

Christian Diangco

Math 381 A

Suppose you play a game using a fair six-sided die.

The game begins at a score of 0.

You roll the die to get your starting score, e.g. rolling a 3 berings the score to 3.

This is how your new score is determined every time your score is equal to 0.

You roll the die again.

If the sum of your roll and your current score is not a prime number, then the sum becomes your score.

If the sum is a prime number less than 26, then the roll is subtracted from your score.

If your score becomes less than 0, then it will be replaced with 0.

You continue rolling the die until your score reaches at least 26.

Now that we know the rules of the game, we’d like to know how many rolls on average it takes to complete it.

We can solve this problem using Markov chains, with each state of the game being stored in a transition matrix *A*.

Because the game goes up to 26, *A* will be 27 by 27 (score ).

Entry *i, j* in *A* represents the probability that the state will change from *i* to *j* after rolling the die.

The probability of the state changing to *j* from current state *i* is based on the sums of *i* with each value on the die.

Note that *j* = 0 is the state that you end up in when the score is less than or equal to 0, and *j* = 26 is the state that you end up in when the score is greater than or equal to 26

Also note that ; this is because 26 is an absorbing state.

Once our score reaches 26, it cannot change to any other score, i.e. the game is over.

Given the die is fair, there is an equal probability () that the die lands on 1, 2, 3, 4, 5, or 6.

However, for some *i*, certain values of *j* are more probable.

For example, at score 1, the following sums are prime: 1+1, 1+2, 1+4, 1+6.

So there is a probability of that the score will become 0.

The following Sage code will be used to construct matrix *A*:

# We will us sympy.isprime() to check if numbers are prime

from sympy import isprime

# This is the score that must be reached to end the game.

max\_score = 26

# Matrix A is the transition matrix for the game.

A = zero\_matrix(QQ, max\_score + 1)

# At score 0, there is a 1/6 chance for the score to change to any of these values: {1,2,3,4,5,6}

A[0,1:7] = 1/6

# Once we reach the max\_score, we want to stay there.

A[max\_score,max\_score] = 1

# new represents the new score given the current score and die value.

new = 0

# This loop fills matrix A.

for current in range(1,max\_score):

less\_than\_0 = 0

greater\_than\_max = 0

for roll in range(1,7):

if isprime(current + roll) and current + roll < max\_score:

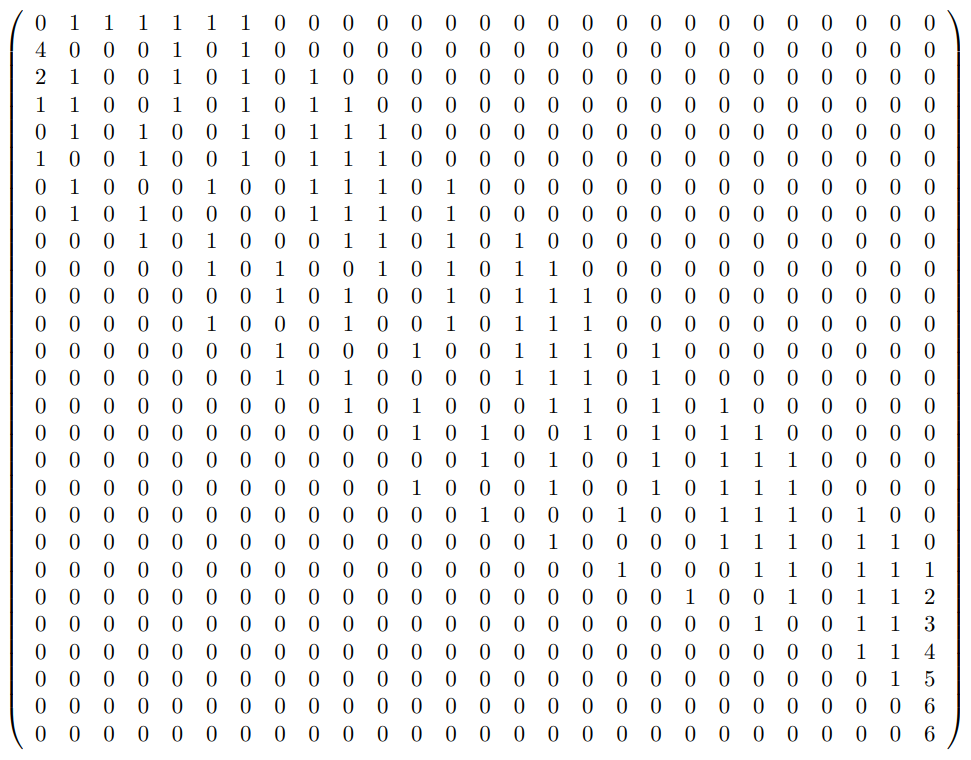
new = current - roll if current - roll > 0 else 0

else:

new = current + roll if current + roll < max\_score else max\_score

A[current,new] += 1/6

This is what matrix 6*A* looks like after running this code:



Looking at *A,* we can see that it is in canonical form.

Therefore, by removing the last row and column of *A,* we can obtain the matrix *Q.*

Using *Q,* we can obtain matrix which can tell us the average number of rolls needed to complete the game.

The sum of the *i*-th row of *N* gives the mean number of steps until absorption when the chain is started in state *i.*

Therefore,we can take the sum of the first row to get the average number of rolls to get at least 26.

We can do this by adding the following lines to our Sage code:

Q = A[:-1,:-1]

N = (matrix.identity(max\_score) - Q).inverse()

sum(N[0])

This outputs the value:

So, on average, it will take about 15.11 rolls for the score to reach at least 26.

What if we want to know the probability that at least half the expected number of rolls are needed to get at least 26?

We can answer this by using matrix *A*.

gives the probabilities for changing scores after *n* - 1rolls.

So we can look at the last entry of the first row into find the probability of reaching 26 in 7 or fewer rolls.

However, we want the probability of requiring more than 7 rolls, which would be the probability of reaching 26 in 7 or fewer rolls subtracted from 1.

We can compute this by adding the following line to our Sage code:

1 - (A\*\*7)[0,-1]

This outputs the value:

So, there is a roughly 89% chance that 26 is reached in after more than 7 rolls.

What about the average number of rolls for different score limits?

How does the number of rolls change as the score increases or decreases?

We can find these the average number of rolls for different score limits the same way as for 26 by changing the max\_score variable in our code.

Let’s create a plot of the average number of rolls for a wide range of score limits using the following Python code:

import matplotlib.pyplot as plt

import numpy as np

# We will us sympy.isprime() to check if numbers are prime

from sympy import isprime

# Score limits that we will use for plotting.

limits = np.array([50,100,500,1000,5000,10000,15000,20000])

# Average number of rolls for each limit in limits.

rolls = np.zeros(len(limits))

for i in range(len(limits)):

# This is the score that must be reached to end the game.

max\_score = limits[i]

# Matrix A is the transition matrix for the game.

A = np.zeros((max\_score + 1, max\_score + 1))

# At score 0, there is a 1/6 chance for the score to change to any of these values: {1,2,3,4,5,6}

A[0,1:7] = 1/6

# Once we reach the max\_score, we want to stay there.

A[max\_score,max\_score] = 1

# new represents the new score given the current score and die value.

new = 0

# This loop fills matrix A.

for current in range(1,max\_score):

less\_than\_0 = 0

greater\_than\_max = 0

for roll in range(1,7):

if isprime(current + roll) and current + roll < max\_score:

new = current - roll if current - roll > 0 else 0

else:

new = current + roll if current + roll < max\_score else max\_score

A[current,new] += 1/6

Q = A[:-1,:-1]

N = np.linalg.inv(np.eye(max\_score) - Q)

rolls[i] = sum(N[0])

fig, ax = plt.subplots()

ax.set\_xlabel("Score limit")

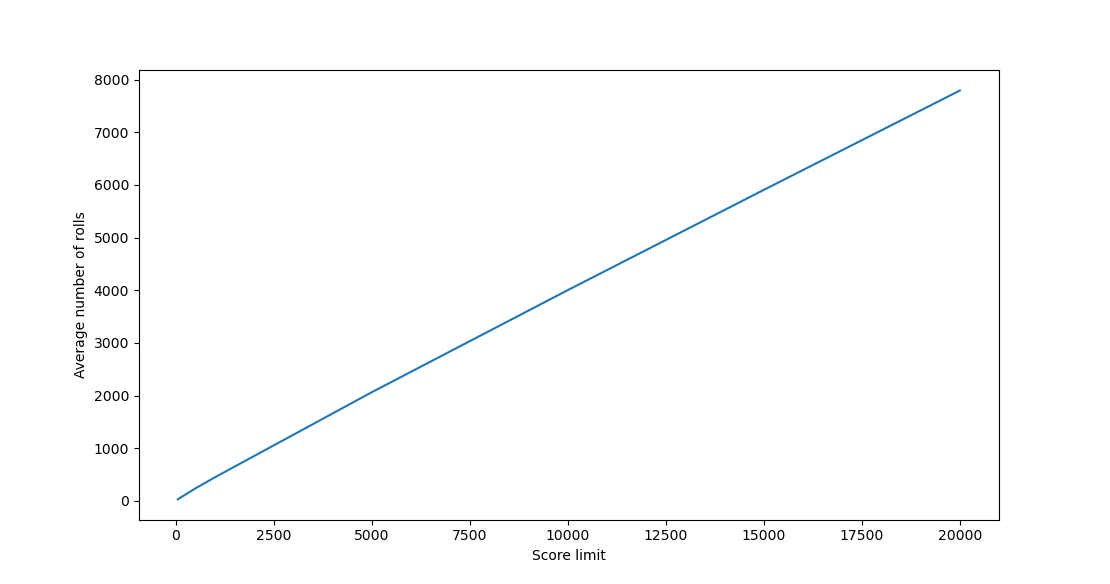
ax.set\_ylabel("Average number of rolls")

ax.plot(limits, rolls)

plt.show()

The code generates a plot of score limits vs average number of rolls for the following limits:

50, 100, 500, 1000, 5000, 10000, 15000, 20000



Looking at the graph, the relationship appears to be linear, however we should verify if this is true.

We can do this by looking at the values for .

If the relationship is linear, then we should see this value approach a single number (i.e. the slope of the line) as the score limit gets increasingly large.

We will calculate the number of rolls and for score limits [20, 1000] with a step count of 20 using the following Sage code:

# We will us sympy.isprime() to check if numbers are prime

from sympy import isprime

for i in range(20,1000+20,20):

# This is the score that must be reached to end the game.

max\_score = i

# Matrix A is the transition matrix for the game.

A = zero\_matrix(QQ, max\_score + 1)

# At score 0, there is a 1/6 chance for the score to change to any of these values: {1,2,3,4,5,6}

A[0,1:7] = 1/6

# Once we reach the max\_score, we want to stay there.

A[max\_score,max\_score] = 1

# new represents the new score given the current score and die value.

new = 0

# This loop fills matrix A.

for current in range(1,max\_score):

less\_than\_0 = 0

greater\_than\_max = 0

for roll in range(1,7):

if isprime(current + roll) and current + roll < max\_score:

new = current - roll if current - roll > 0 else 0

else:

new = current + roll if current + roll < max\_score else max\_score

A[current,new] += 1/6

Q = A[:-1,:-1]

N = (matrix.identity(max\_score) - Q).inverse()

rolls = sum(N[0])

print(max\_score, float(rolls), float(rolls/max\_score), sep=" ")

The code gives us the following output:

20 12.136372927769319 0.6068186463884659

40 22.360452890335413 0.5590113222583853

60 33.040304462852156 0.550671741047536

80 44.54756129260215 0.5568445161575268

100 53.09846716219185 0.5309846716219185

120 65.05222843847929 0.542101903653994

140 74.76382567551995 0.534027326253714

160 83.60985471473774 0.5225615919671108

180 92.46156310464178 0.5136753505813433

200 104.45487565320101 0.5222743782660051

220 111.08454636310776 0.5049297561959444

240 121.63332943549511 0.506805539314563

260 130.5476644382812 0.502106401685697

280 140.08241500979264 0.500294339320688

300 149.11328840196566 0.4970442946732188

320 159.03006573474545 0.4969689554210795

340 166.25937825040702 0.48899817132472656

360 175.80438365519936 0.48834551015333155

380 183.83658069234565 0.48378047550617276

400 192.33701199382236 0.48084252998455584

420 200.7312060251182 0.4779314429169481

440 210.82187356072092 0.4791406217289112

460 219.35744904694698 0.47686401966727604

480 229.21778045396462 0.4775370426124263

500 237.78174606783398 0.475563492135668

520 245.36691348156268 0.47185944900300514

540 253.24133728628865 0.46896543941905305

560 261.22584956052555 0.4664747313580813

580 270.7483112542104 0.46680743319691453

600 278.5788556173925 0.4642980926956541

620 289.8440078983301 0.4674903353198873

640 296.47793702171583 0.463246776596431

660 306.72828295803805 0.464739822663694

680 315.8994368466279 0.46455799536268816

700 323.25523344376535 0.4617931906339505

720 331.3716605357018 0.46023841741069693

740 339.3857730921934 0.45862942309755866

760 347.8848296402136 0.4577431968950179

780 356.63497266333957 0.45722432392735846

800 363.9393195075387 0.4549241493844234

820 371.8348018686482 0.453457075449571

840 382.9993890089368 0.45595165358206763

860 391.8524947305293 0.45564243573317365

880 399.5146616205281 0.45399393365969104

900 408.6676785616702 0.45407519840185573

920 417.019849810656 0.45328244544636526

940 424.5149211680402 0.4516116182638725

960 432.707036443885 0.4507364962957135

980 440.97920771490806 0.44997878338255926

1000 449.0158898326911 0.4490158898326911

The leftmost value is the score limit, the middle value is the average number of rolls, and the rightmost value is .

It appears that just continually decreases rather than converging at a value, so the relationship does not appear to be linear.

Since the value is decreasing, this means the score limit is increasing at a faster rate than the average number of rolls.

This could mean that there is a relationship where for some constant *a,* where .

This implies that .

If there is such a value *a*, then as the score limit becomes increasingly large, will converge to a single value.

We will edit our code above so that float(rolls/max\_score)is replaced by float(log(rolls)/log(max\_score)).

This gives us the following output:

20 12.136372927769319 0.8332543574276919

40 22.360452890335413 0.8423408629153536

60 33.040304462852156 0.8542828065475229

80 44.54756129260215 0.8663930477899136

100 53.09846716219185 0.8625409920604111

120 65.05222843847929 0.8721039504629291

140 74.76382567551995 0.8730567258971713

160 83.60985471473774 0.8721201585688738

180 92.46156310464178 0.871717819758182

200 104.45487565320101 0.8774021721626154

220 111.08454636310776 0.8733067958158443

240 121.63332943549511 0.875994768106511

260 130.5476644382812 0.8761045368326913

280 140.08241500979264 0.87709236299926

300 149.11328840196566 0.8774363966592226

320 159.03006573474545 0.8787814132937775

340 166.25937825040702 0.877268278678881

360 175.80438365519936 0.8782331900332029

380 183.83658069234565 0.8777604201515995

400 192.33701199382236 0.8777902402309985

420 200.7312060251182 0.8777720449009317

440 210.82187356072092 0.8791213435490569

460 219.35744904694698 0.8792209186729365

480 229.21778045396462 0.8802819641821709

500 237.78174606783398 0.8804019711234651

520 245.36691348156268 0.8799017149483006

540 253.24133728628865 0.8796443007471573

560 261.22584956052555 0.8794944619931261

580 270.7483112542104 0.8802710843689099

600 278.5788556173925 0.8800630094075176

620 289.8440078983301 0.8817403144605744

640 296.47793702171583 0.8809101351153426

660 306.72828295803805 0.881970233600268

680 315.8994368466279 0.8824504807300706

700 323.25523344376535 0.8820594332650444

720 331.3716605357018 0.8820518323371042

740 339.3857730921934 0.8820108935502233

760 347.8848296402136 0.8821936613700312

780 356.63497266333957 0.8824828644035915

800 363.9393195075387 0.8821734738432818

820 371.8348018686482 0.8821256831544876

840 382.9993890089368 0.8833623017679986

860 391.8524947305293 0.8836680813211735

880 399.5146616205281 0.883527944157762

900 408.6676785616702 0.8839390410466871

920 417.019849810656 0.8840567814470511

940 424.5149211680402 0.8838815850364117

960 432.707036443885 0.8839551306764791

980 440.97920771490806 0.8840582576961683

1000 449.0158898326911 0.8840872367149535

It appears that converges to ~ .88.

So, roughly,

To verify our computations for the average number of rolls are correct, we will calculate the expected value for one limit and run simulations of the game with that limit.

From these simulations, we will calculate the average number of rolls and if our computations are correct, the value from the simulations should be about the same as the value from using Markov chains.

We will choose to use a score limit of 99, arbitrarily.

By changing max\_score in our code to 99, the outputted value for sum(N[0])is

We will check if this computed value is accurate by running 5 simulations of 100,000 games where the score limit is 99.

We will use this Sage code to run the simulations:

from sympy import isprime

from random import randint

# This is the score that must be reached to end the game.

max\_score = 99

# Run 5 simulations

for i in range (5):

# This is the sum of the number of rolls for each game.

sum = 0

# Play 100000 games.

for i in range (100000):

# This keeps track of the current score.

score = 0

# This keeps track of the number of rolls.

rolls = 0

# Rolls a die until max\_score is reached.

while score < max\_score:

roll = randint(1,6)

rolls += 1

if (score <= 0):

score = roll

elif (score + roll < max\_score and isprime(score + roll)):

score -= roll

else:

score += roll

sum += rolls

mean = sum / 100000

print(mean, float(mean), sep=" ")

Running this code prints the following output:

5271771/100000 52.71771

2633319/50000 52.66638

1319681/25000 52.78724

211043/4000 52.76075

2638259/50000 52.76518

Each row in the output represents a simulation of 100,000 games.

The value on the left is the exact value of the mean number of rolls for a simulation.

The value on the right is the decimal approximation.

We can see that each of the means are very close to our expected value of rolls for a score limit of 99.

Thus, we can conclude that our computations have little error, if any.