Modeling Foot Traffic

TABLE OF CONTENTS

The following document contains screenshots of code cells and the output for this assignment. Hyperlinks are added as bookmarks to take the reader directly to the figure/section.

Site Requirements:

Given that we were in two different countries, we are including two site maps and will be conducting our analysis on one of the locations. The chosen network for analysis (Fig. 2) is in the section "Network Selection" and the other network/site map is in Appendix I.

Location 1:

White Field Road and Gachibowli-Miyapur Road in HITEC city Hyderabad (Fig. 1).

Location 2:

Bahria Town Food Street, Islamabad, Pakistan (Fig. 3).

Model Analysis Report:

Network Selection:

We selected an intersection in HITEC city Hyderabad (Fig . 2). The intersection represents the main exit from the residential area of Ashok Nagar neighborhood onto the busy highway that is G-M Road. As such, there is significant foot traffic in the area that local small businesses monetize. Because the space is right next to a busy road, shops here are mostly catering to drivers and commuters visiting for a quick pit stop. For us, this means we can observe customers coming and leaving any given node in a matter of minutes.

Nodes:

- Desi Bytes – food stand
- Chat Bandar – snack stand
- Dimmy Pan Palace – paan shop (dessert/drug/digestive)

- Vellaniki Foods – sweets shop
- Khushi Juice Point – juice stand

The nodes' relatively diverse specialties make them less competitive and, rather, allow for a mutually beneficial flow of customers. For example, one might have a meal at DB, chew some paan for digestion at DPP, and then grab a coconut on the way from KJP. For our purposes, this means that our system will be a network with many paths between the nodes that people can use in navigating the intersection.

We chose this location because of logistics related to Ramadan. Data was collected on April 3rd at 2 PM (2:30 IST), thus the foot traffic at the location in Pakistan was not as representative at that time. Additionally, it was easier to observe the foot traffic in this open area with fast food stalls compared to dining areas such as Tuscany Courtyard.

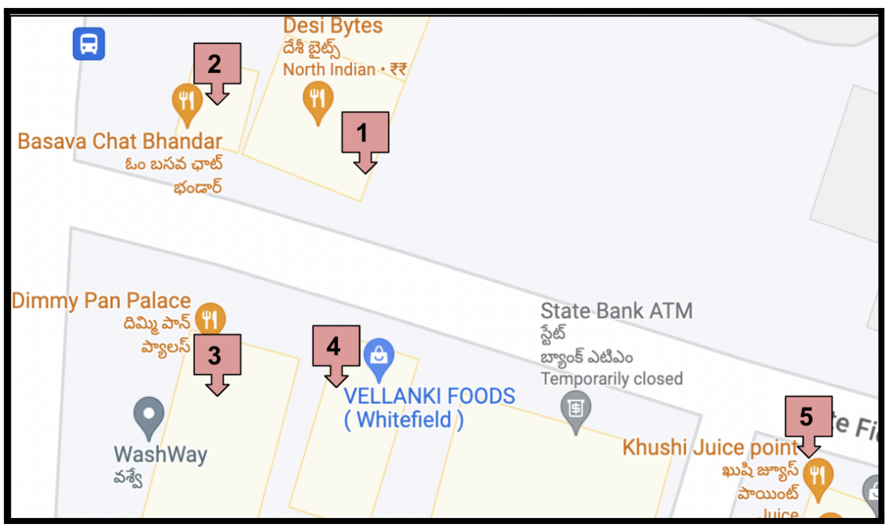

Fig. 1 shows a Google Maps image of the food junction in Ashok Nagar, Hyderabad, India.

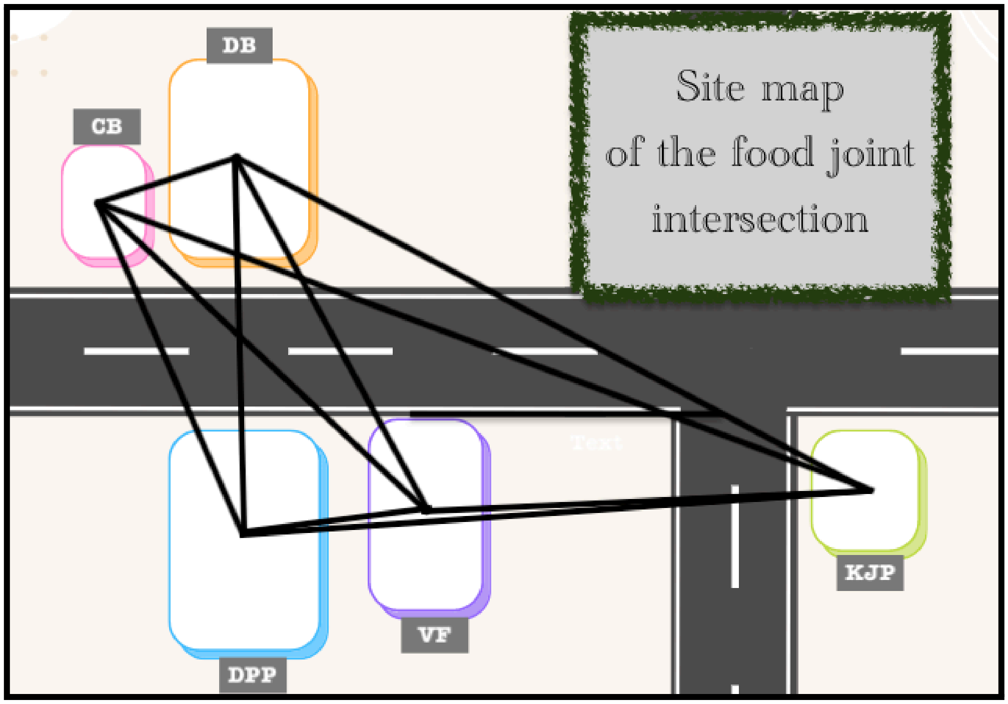

Fig. 2 shows the network of the food junction in Ashok Nagar, Hyderabad, India with the following food stalls: Desi Bytes (DB), Chat Bandar (CB), Dimmy Pan Palace (DPP), Vellanki Foods (VF), and Khushi Juice Point (KJP).

Data Collection:

Data was collected by observing foot traffic from near the center of the five nodes. From regular interaction with those vendors, we expected that food preparation will take between 2 to 10 minutes so we collected data in intervals of 5 minutes for 20 minutes. We counted people based on whether or not they joined or left that location within that time interval.

Assumptions:

1. The area selected is popular to get a large sample size.

2. Regardless of the food stall/restaurant, the time taken to prepare food will be approximately the same and short (between 2-5 minutes) as for food stalls in Fig .1 snacks were offered that take less time and in restaurants (Fig. 2), people were picking up food for deliveries or themselves due to Ramadan.
3. People will not eat the food at the same location. This works for Location 2 as well as with Ramadan, the majority of the people coming to fetch food were not dining in the restaurant.

Limitations:

1. The network is not a closed system as there is no entrance or exit and people from other food areas can also come to these five nodes.
2. Not every person stays at the same food stall as they might be checking their options and might visit more than one stall.

We will be using Table 2 for our analysis; however, for proof of data collection by both students, Table 3 is included to show the data of the second location in Appendix I.

T/min	Desi Bytes	Chat Bandar	Dimmy Pan Palace	Vellanki Foods	Khushi Juice Point
0	4	6	1	0	0
5	2	3	5	0	4
10	3	0	12	0	0
15	0	1	8	1	1
20	0	2	5	1	0

Table 2: The data collected at the Ashok Chowk food junction, Hyderabad, India.

Probability Distribution Matrix:

Each table represents a matrix that shows the transition from the current state to the next state with five-minute time intervals.

	Current State				
Next State	**Desi Bytes**	**Chat Bandar**	**Dimmy Pan Palace**	**Vellanki Foods**	**Khushi Juice Point**
Desi Bytes	4	0	0	0	0
Chat Bandar	0	6	0	0	0
Dimmy Pan Palace	0	0	1	0	0
Vellanki Foods	0	0	0	0	0
Khushi Juice Point	0	0	0	0	0

Table 3: The distribution of people at the start (0 minutes)

	Current State				
Next State	**Desi Bytes**	**Chat Bandar**	**Dimmy Pan Palace**	**Vellanki Foods**	**Khushi Juice Point**
Desi Bytes	2	0	0	0	0
Chat Bandar	0	3	0	0	0
Dimmy Pan Palace	0	0	5	0	0
Vellanki Foods	0	0	0	0	0
Khushi Juice Point	0	0	0	0	4

Table 4: The distribution of people at 5 minutes.

	Current State				
Next State	**Desi Bytes**	**Chat Bandar**	**Dimmy Pan Palace**	**Vellanki Foods**	**Khushi Juice Point**
Desi Bytes	3	0	0	0	0
Chat Bandar	0	0	0	0	0
Dimmy Pan Palace	0	0	12	0	0
Vellanki Foods	0	0	0	0	0
Khushi Juice Point	0	0	0	0	0

Table 5: The distribution of people at 10 minutes.

	Current State				
Next State	**Desi Bytes**	**Chat Bandar**	**Dimmy Pan Palace**	**Vellanki Foods**	**Khushi Juice Point**
Desi Bytes	0	0	0	0	0
Chat Bandar	0	1	0	0	0
Dimmy Pan Palace	0	0	8	0	0
Vellanki Foods	0	0	0	1	0
Khushi Juice Point	0	0	0	0	1

Table 6: The distribution of people at 15 minutes.

	Current State				
Next State	**Desi Bytes**	**Chat Bandar**	**Dimmy Pan Palace**	**Vellanki Foods**	**Khushi Juice Point**
Desi Bytes	0	0	0	0	0
Chat Bandar	0	2	0	0	0
Dimmy Pan Palace	0	0	5	0	0
Vellanki Foods	0	0	0	1	0
Khushi Juice Point	0	0	0	0	0

Table 7: The distribution of people at 20 minutes.

For each table, we will use the following methodology to obtain people left and the probability of staying at that location:

Location

People Left: $People\ at\ the\ start\ -\ People\ after\ the\ time\ interval$

Probability of staying (if people leave): $\frac{People\ at\ 0\ minutes - People\ Left}{People\ at\ 0\ minutes}$

In the case people join, the probability of staying will be 1 as no person left the location.

The math and calculations are shown in Appendix II.

$$\begin{bmatrix} \frac{7}{16} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{16} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

Fig. 5 shows a 5x5 matrix with the diagonal showing probabilities of staying at the location.

The above matrix shows only the probability distribution of foot traffic where the people stay at a given place over all the time stamps. However, to form a Markov matrix, we need to ensure the columns equal to 1.

$$\begin{bmatrix} \frac{7}{16} & 0 & 0 & \frac{1}{8} & \frac{1}{2} \\ 0 & \frac{1}{4} & 0 & \frac{1}{8} & 0 \\ \frac{5}{16} & \frac{3}{4} & 1 & \frac{1}{8} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{4} & 0 & 0 & \frac{1}{8} & \frac{1}{2} \end{bmatrix}$$

Fig. 6 shows a 5x5 Markov matrix of the selected network.

The above matrix shows a Markov matrix with the justification given below for balancing the probability distribution in columns.

Column 1 for Desi Bytes:

$$\boxed{\begin{matrix} \frac{7}{16} \\ 0 \\ \frac{5}{16} \\ 0 \\ \frac{1}{4} \end{matrix}}$$

→ **1a.** Probability of remaining at Desi Bytes **(DB)**

→ **2a.** Probability of moving to Dimmy Pan Palace **(DPP)**

→ **3a.** Probability of moving to Khushi Juice Point **(KJP)**

We made some assumptions in determining the probabilities for 2a and 3a above.

1. The probability of moving from DB to DPP $\frac{5}{16}$ represents a higher probability than moving from DB to KJP $\frac{1}{4}$ as DB is closer to DPP than KJP as shown in Fig. 2 so more people will move to DPP than KJP.
2. Additionally, traffic is moving from DB to only these two locations as we observed from Table 1 - 5 that most people leaving DB coincided with more people joining DPP and KJP. This assumption has been utilized for each of the five columns in the Markov matrix above.

Column 2 for Chat Bandar:

$$\begin{array}{|c|l}
\cline{1-1}
0 & \\
\frac{1}{4} & \rightarrow \textbf{1b.} \text{ Probability of remaining at Chat Bandar } \textbf{(CB)} \\
\frac{3}{4} & \rightarrow \textbf{2b.} \text{ Probability of moving to Dimmy Pan Palace } \textbf{(DPP)} \\
0 & \\
0 & \\
\cline{1-1}
\end{array}$$

The two conditions for column 1 applies here too:

1. CB and DPP are closer to each other (Fig. 2).
2. People leaving CB coincided with people joining DPP.

Column 3 for Dimmy Pan Palace:

0	
0	
1	→ **1c.** Probability of remaining at Dimmy Pan Palace **(DPP)**
0	
0	

No balancing was needed here.

Column 4 for Vellanki Foods:

$\frac{1}{8}$	→ **1d.** Probability of moving to Desi Bytes **(DB)**
$\frac{1}{8}$	→ **2d.** Probability of moving to Chat Bandar **(CB)**
$\frac{1}{8}$	→ **3d.** Probability of moving to Dimmy Pan Palace **(DPP)**
$\frac{1}{2}$	→ **4d.** Probability of remaining at Vellanki Foods **(VF)**
$\frac{1}{8}$	→ **5d.** Probability of moving to Khushi Juice Point **(KJP)**

For half the duration of time, there was no traffic at VF, and the overall traffic was one for the rest of the duration which makes VF difficult to analyze as it might have multiple confounding factors. Therefore, in order to cater to that randomization at any given time, we assumed that movement of foot traffic to other food locations will be equally likely due to the lack of information.

Column 5 for Khushi Juice Point:

$$\begin{array}{|c|l} \cline{1-1} \frac{1}{2} & \rightarrow \textbf{1e.}\ \text{Probability of remaining at Desi Bytes } \textbf{(DB)} \\ 0 & \\ 0 & \\ 0 & \\ \frac{1}{2} & \rightarrow \textbf{2e.}\ \text{Probability of moving to Khushi Juice Point } \textbf{(KJP)} \\ \cline{1-1} \end{array}$$

The two conditions for column 1 applies here too:

1. CB and DPP are closer to each other (Fig. 2).
2. People leaving KJP coincided with people joining DB.
3. Even though VP is closer to KJP, there was no significant foot traffic at VP so we ignored it.

Analysis of Model:

Appendix III has calculations for all eigenvectors, eigenvalues and stationary distributions.

Because we found it unrealistic to track where each of the individual agents ends up in our system after transit, due to it being an open, busy space, we calculated the probabilities of going from all current states (rows) to terminal states (columns) based on the number in transit and differences in the distribution of agents at different time intervals as described in Appendix II. This means that our matrix is in terms of probabilities (columns add up to 1) which, paired with the normalized vectors we used for our initial states, v_0, means the result of their multiplication is going to readily provide us with a probability distribution, as can be seen in Appendix III. We then calculated the eigenvalues and found the eigenvectors using NumPy's 'linalg.eig()' function which has the advantage of optimized speed, stability, and consistency with common user practice (Brownlee, 2023) compared to our other available tools and resources. We found the vector $\langle 0, 0, 1, 0, 0 \rangle$ corresponding to the eigenvalue 1 which represents our stationary state. This method comes from the theory of linear algebra. Because for v, an eigenvector of a square matrix M, the property $M \times v = \lambda \times v$ must be satisfied (Kuttler, 2023), it follows that for our λ value of 1, the vector will remain unchanged after the multiplication (stationary state is reached).

To test the results of multiplying our increasing Markov matrix powers with initial distribution vectors, we set 'for loops' plotting our probabilities for each location as the powers increase on the y-axis against the powers of our matrix, our time steps, on the x-axis. As can be seen, regardless of the initial distribution, the system converges into a stationary state approaching the distribution of the stationary found from the eigenvectors. This behavior was expected as it can be explained by the convergence theorem for Markov chains (Suhov, 2009). We could have expected other results if we had an eigenvalue of -1, which would cause an alternating behavior corresponding to $A \times v = -v$ or a value between -1 and 1, a case in which the Markov process could still converge, but without necessarily having a single stationary state.

In our case, the behavior of the Markov process means that, in time, all of the customers will gather at Dimmy Pan Palace, the stationary state given by our eigenvector (probability distribution of finding the customers in our system). This convergence will happen at a rate determined by our eigenvalue, λ. Despite the oversimplifications, this insight could be useful for the owners of these businesses in managing their resources and activity to maximize their profits.

Appendix I (Site Maps and Data for the second student)

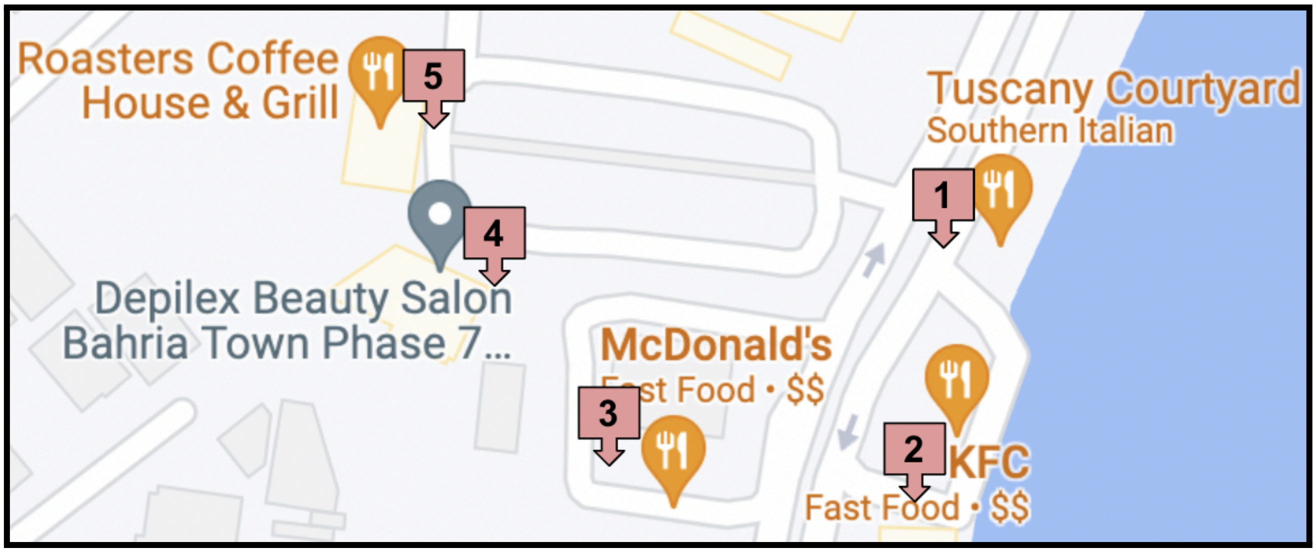

Fig. 3 shows a Google Maps image of Bahria Food Court, Islamabad, Pakistan.

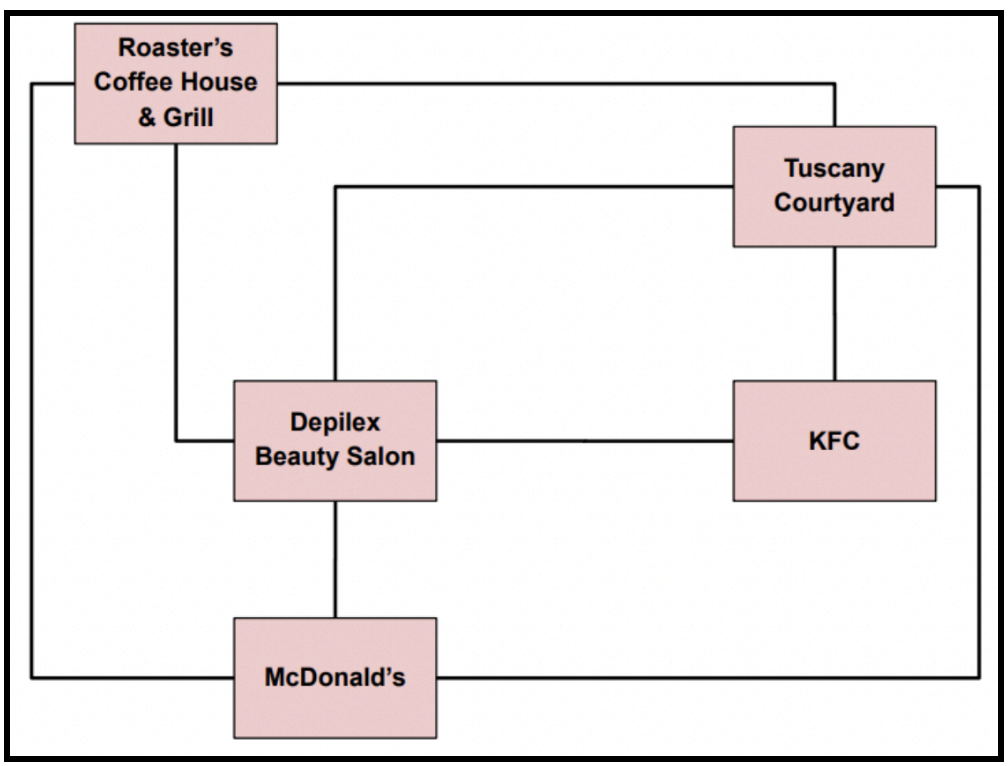

Fig. 4 shows a site map of Bahria Food Court, Islamabad, Pakistan.

T/min	Tuscany Courtyard	KFC	McDonalds	Depilex Beaty Salon	Roaster's Coffee Houses
0	0	2	1	0	0
5	2	2	0	0	2
10	3	0	2	0	0
15	0	0	0	0	1
20	0	1	0	0	0

Table 2: The data collected at Bahria Town Food Street, Islamabad, Pakistan.

Appendix II (Probability Distribution Calculations)

Using Table 3 and Table 4:

Desi Bytes	Chat Bander	Dimmy Pan Palace	Vellanki Foods	Khushi Juice Point
People Left: $4 - 2 = 2$ **Probability of staying:** $\frac{4-2}{4} = \frac{1}{2}$	**People Left:** $6 - 3 = 3$ **Probability of staying:** $\frac{6-3}{6} = \frac{1}{2}$	**People Left:** $1 - 5 = -4$ so 4 people joined. Therefore, **Probability of staying:** 1	**People Left:** $0 - 0 = 0$ **Probability of staying:** 0	**People Left:** $0 - 4 = -4$ so 4 people joined. Therefore, **Probability of staying:** 1

Using Table 3 and Table 5:

Desi Bytes	Chat Bander	Dimmy Pan Palace	Vellanki Foods	Khushi Juice Point
People Left: $4 - 3 = 1$ **Probability of staying:** $\frac{4-1}{4} = \frac{3}{4}$	**People Left:** $6 - 0 = 6$ **Probability of staying:** $\frac{6-6}{6} = 0$	**People Left:** $1 - 12 = -11$ so 11 people joined. Therefore, **Probability of staying:** 1	**People Left:** $0 - 0 = 0$ **Probability of staying:** 0	**People Left:** $0 - 0 = 0$ Therefore, **Probability of staying:** 0

Using Table 3 and Table 6:

Desi Bytes	Chat Bander	Dimmy Pan Palace	Vellanki Foods	Khushi Juice Point
People Left: $4 - 0 = 4$ **Probability of staying:** $\frac{4-4}{4} = 0$	**People Left:** $6 - 1 = 5$ **Probability of staying:** $\frac{6-5}{6} = \frac{1}{6}$	**People Left:** $1 - 8 = -8$ so 8 people joined. Therefore, **Probability of staying:** 1	**People Left:** $0 - 1 = -1$ so 1 person joined. Therefore, **Probability of staying:** 1	**People Left:** $0 - 1 = -1$ so 1 person joined. Therefore, **Probability of staying:** 1

Using Table 3 and Table 7:

Desi Bytes	Chat Bander	Dimmy Pan Palace	Vellanki Foods	Khushi Juice Point
People Left: $4 - 2 = 2$ **Probability of staying:** $\frac{4-2}{4} = \frac{1}{2}$	**People Left:** $6 - 2 = 4$ **Probability of staying:** $\frac{6-4}{6} = \frac{1}{3}$	**People Left:** $1 - 2 = -1$ so 1 person joined. Therefore, **Probability of staying:** 1	**People Left:** $0 - 1 = -1$ so 1 person joined. Therefore, **Probability of staying:** 1	**People Left:** $0 - 0 = 0$ **Probability of staying:** 0

Average probabilities of staying:

We will sum up all the probabilities of staying for each time stamp and then take an average.

Desi Bytes:

Prob of staying at Desi Bytes: $\frac{\frac{1}{2}+\frac{3}{4}+0+\frac{1}{2}}{4} = \frac{7}{16} = 0.4375$

Chat Bander:

Prob of staying at Chat Bander: $\frac{\frac{1}{2}+0+\frac{1}{6}+\frac{1}{3}}{4} = \frac{1}{4} = 0.25$

Dimmy Pan Palace:

Prob of staying at Dimmy Pan Palace: $\frac{1+1+1+1}{4} = 1$

Vellanki Foods:

Prob of staying at Vellanki Foods: $\frac{0+0+1+1}{4} = \frac{1}{2} = 0.5$

Khushi Juice Point:

Prob of staying at Khushi Juice Point: $\frac{1+0+1+0}{4} = \frac{1}{2} = 0.5$

Appendix III (Eigenvectors and Long Term Probability Behavior)

In [128]:
```
Mark = matrix([[7/16,0,0,1/8, 1/2],
[0,1/4,0,1/8,0],[5/16,3/4,1,1/8,0],[0,0,0,1/2,0],[1/4,0,0,1/8,1/2]]).transpose()
```

In [129]:
```
Mark
```

Out[129]:
```
[7/16    0 5/16    0  1/4]
[   0  1/4  3/4    0    0]
[   0    0    1    0    0]
[ 1/8  1/8  1/8  1/2  1/8]
[ 1/2    0    0    0  1/2]
```

In [28]:
```
import numpy as np
from numpy.linalg import eig
from sympy import simplify

eigenvalues, eigenvectors = eig(Mark)
print('\n eigenvalues =',eigenvalues, '\n eigenvectors = \n', eigenvectors)
```

```
 eigenvalues = [1.         0.82368177 0.11381823 0.25       0.5       ]
 eigenvectors =
 [[ 0.          0.45942097  0.80492654  0.         -0.34749779]
 [ 0.          0.          0.          0.70710678  0.34749779]
 [ 1.         -0.81426098 -0.28384644 -0.70710678 -0.47780947]
 [ 0.          0.          0.          0.          0.69499559]
 [ 0.          0.35484001 -0.5210801   0.         -0.21718612]]
```

In [101]:
```
# we take the eigenvector related to the eigenvalue equal to 1
stationary = vector([0,0,1,0,0])
# we check that multiplying the markov matrix by
# the stationary leaves the matrix unchanged
assert Mark*stationary == stationary
```

We can check that this approach makes sense with the formula from the theory of linear algebra. We know that an eigenvector is a vector that stays in span after multiplication with a matrix (or other linear transformation), except for a scaling factor. e.g. we have a square matrix, M. v is an eigenvector if there is a value λ, a scalar, that satisfies the property: $M \times v = \lambda \times v$

Because our λ is 1, the equation becomes: $M \times v = v$ which means v remains unchanged after multiplication. Thus v is a stationary distribution

```
In [97]:
# if the main flow of customers comes from a direction
# (residentiala area or main road)
# we make an assumption that the number of people is reduced exponentially
# as the crowd progresses through the road
# to simulate this we will set an initial distribution starting with half
# on the first node, reducing half of what's left for each consecutive node
people = 100
nodes = 5
distribution = []
for i in range(nodes):
    people = percent_people * 0.5^(i+1)
    distribution.append(people)

#normalize
distribution = [x/np.sum(distribution) for x in distribution]
rev_distribution = sorted(distribution)
```

```
In [86]:
# no assumptions
initial_state1 = vector([1/5,1/5,1/5,1/5,1/5])
# first one coming from the residential area
initial_state2 = vector(distribution)
# first one coming from the main road
initial_state3 = vector(rev_distribution)
# the start on one only
initial_state4 = vector([1,0,0,0,0])
```

```
import matplotlib.pyplot as plt

db = []
cb = []
dpp = []
vf = []
kjp = []
for n in range(101):
    result = (Mark^n)*initial_state1
    db.append(result[0])
    cb.append(result[1])
    dpp.append(result[2])
    vf.append(result[3])
    kjp.append(result[4]
              )
x = list(range(101))

plt.plot(x, db, label = 'Desi Bytes', color = 'darkmagenta')
plt.plot(x, cb, label = 'Chat Bandar', color = 'mediumspringgreen')
plt.plot(x, dpp, label = 'Dimmy Pan Palace', color = 'firebrick')
plt.plot(x, vf, label = 'Vellanki foods', color = 'royalblue')
plt.plot(x, kjp, label = 'Khushi Juice Point', color = 'darkgoldenrod')
plt.title('Long Term Behavior of Probabilities')
plt.xlabel('Powers of Markov Matrix')
plt.ylabel('Probability')
plt.legend()
plt.show()
```

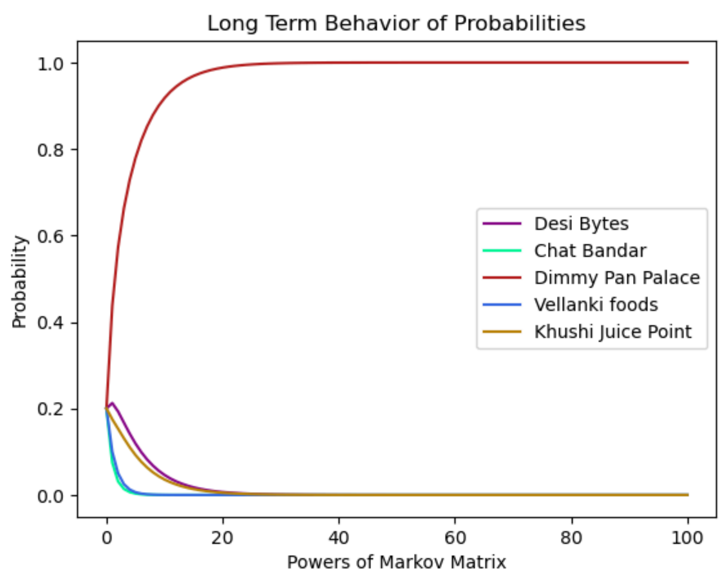

Fig. 7 shows a graph of evolving foot traffic for the first initial state.

```python
import matplotlib.pyplot as plt

db = []
cb = []
dpp = []
vf = []
kjp = []
for n in range(101):
    result = (Mark^n)*initial_state2
    db.append(result[0])
    cb.append(result[1])
    dpp.append(result[2])
    vf.append(result[3])
    kjp.append(result[4]
              )
x = list(range(101))

plt.plot(x, db, label = 'Desi Bytes', color = 'darkmagenta')
plt.plot(x, cb, label = 'Chat Bandar', color = 'mediumspringgreen')
plt.plot(x, dpp, label = 'Dimmy Pan Palace', color = 'firebrick')
plt.plot(x, vf, label = 'Vellanki foods', color = 'royalblue')
plt.plot(x, kjp, label = 'Khushi Juice Point', color = 'darkgoldenrod')
plt.title('Long Term Behavior of Probabilities')
plt.xlabel('Powers of Markov Matrix')
plt.ylabel('Probability')
plt.legend()
plt.show()
```

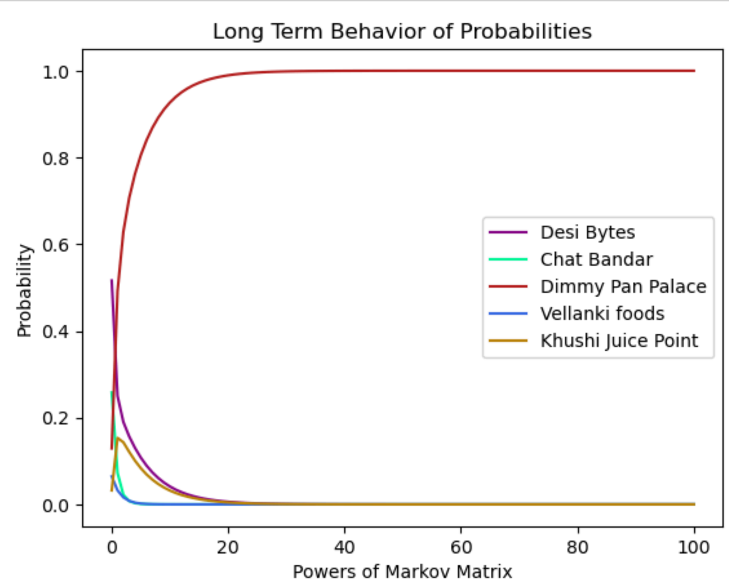

Fig. 8 shows a graph of evolving foot traffic for the second initial state.

```
import matplotlib.pyplot as plt

db = []
cb = []
dpp = []
vf = []
kjp = []
for n in range(101):
    result = (Mark^n)*initial_state3
    db.append(result[0])
    cb.append(result[1])
    dpp.append(result[2])
    vf.append(result[3])
    kjp.append(result[4]
              )
x = list(range(101))

plt.plot(x, db, label = 'Desi Bytes', color = 'darkmagenta')
plt.plot(x, cb, label = 'Chat Bandar', color = 'mediumspringgreen')
plt.plot(x, dpp, label = 'Dimmy Pan Palace', color = 'firebrick')
plt.plot(x, vf, label = 'Vellanki foods', color = 'royalblue')
plt.plot(x, kjp, label = 'Khushi Juice Point', color = 'darkgoldenrod')
plt.title('Long Term Behavior of Probabilities')
plt.xlabel('Powers of Markov Matrix')
plt.ylabel('Probability')
plt.legend()
plt.show()
```

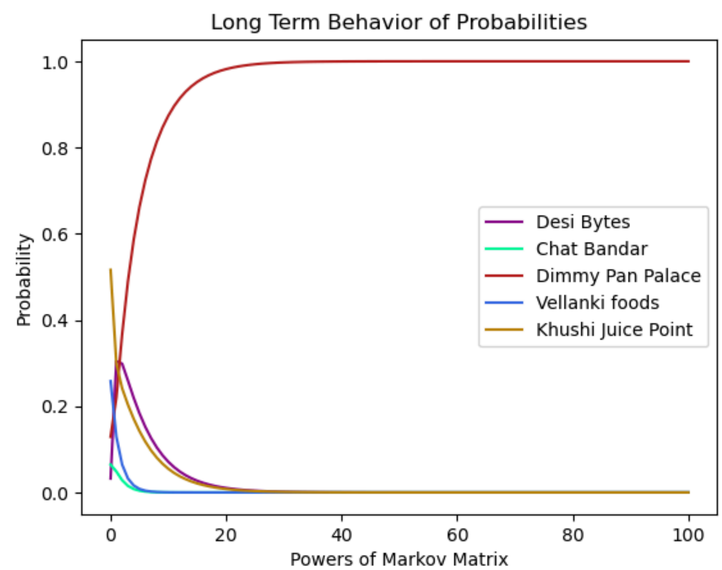

Fig. 9 shows a graph of evolving foot traffic for the third initial state.

```python
import matplotlib.pyplot as plt

db = []
cb = []
dpp = []
vf = []
kjp = []
for n in range(101):
    result = (Mark^n)*initial_state4
    db.append(result[0])
    cb.append(result[1])
    dpp.append(result[2])
    vf.append(result[3])
    kjp.append(result[4]
              )
x = list(range(101))

plt.plot(x, db, label = 'Desi Bytes', color = 'darkmagenta')
plt.plot(x, cb, label = 'Chat Bandar', color = 'mediumspringgreen')
plt.plot(x, dpp, label = 'Dimmy Pan Palace', color = 'firebrick')
plt.plot(x, vf, label = 'Vellanki foods', color = 'royalblue')
plt.plot(x, kjp, label = 'Khushi Juice Point', color = 'darkgoldenrod')
plt.title('Long Term Behavior of Probabilities')
plt.xlabel('Powers of Markov Matrix')
plt.ylabel('Probability')
plt.legend()
plt.show()
```

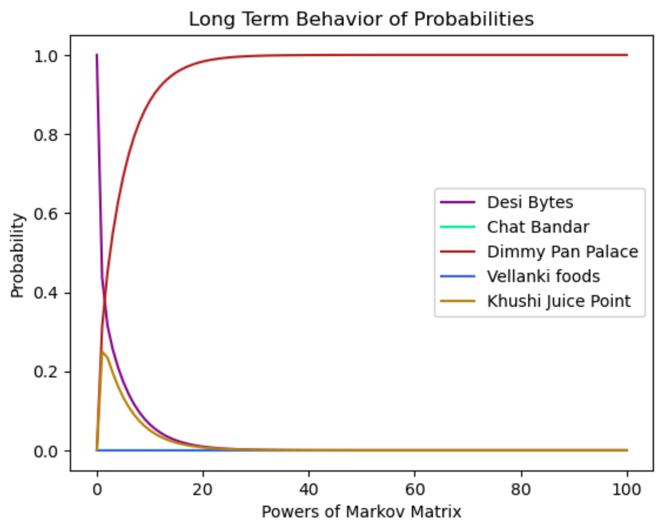

Fig. 10 shows a graph of evolving foot traffic for the fourth initial state.

```
In [125]: # we can verify that no matter the initial distribution
          # we will reach same stationary distribution
          print(((Mark^100)*initial_state1).n())
          np.round_(((Mark^100)*initial_state1).n(), decimals = 8)

(1.21165857636948e-9, 7.88860905221012e-32, 0.999999997852500, 1.57772181044202e-31, 9
.35840911223729e-10)

Out[125]: array([0., 0., 1., 0., 0.])
```

AI Policy

None of our analysis or data collection processes required the use of AI, hence we did not use it for this assignment.

References

Brownlee, J. (2023, March 14). *What is Blas and LAPACK in NumPy*. Super Fast Python. Retrieved April 6, 2023, from https://superfastpython.com/what-is-blas-and-lapack-in-numpy/

Google Maps. (2023). Bahria Town Food Street. https://www.google.com/maps/place/Bahria+Food+St,+Bahria+Intellectual+Village+Rawalpindi,+Punjab/@33.5248931,73.0963547,17z/data=!4m6!3m5!1s0x38dfed4e52afa9db:0xb28e5ab5bc96dc3c!8m2!3d33.5273527!4d73.0978353!16s%2Fg%2F11j4lrnm2q

Google Maps. (2023). Desi Bytes, Ashok Chowk. https://www.google.com/search?q=desi+bytes+hyderabad&biw=1440&bih=789&tbm=lcl&sxsrf=APwXEdcedTEfBZoS7JxCIvznMnhXutxNRw%3A1680791841037&ei=IdkuZM_pAYTxkwXMzYW4Dw&oq=desi+b&gs_lcp=Cg1nd3Mtd2l6LWxvY2FsEAMYADIECCMQJzIICAAQigUQkQIyCggAEIoFELEDEEMyBQgAEIAEMgUIABCABDIFCAAQgAQyBQgAEIAEMgUIABCABDIFCAAQgAQyBQgAEIAEOgsIABCABBCxAxCDAToLCAAQigUQsQMQgwE6BAgAEAM6BwgAEIoFEEM6CggAEIoFEMkDEEM6CAgAEIoFEJIDOg0IABCABBCxAxCDARAKOgcIABCABBAKOgoIABCABBDJA

xAKUABY3BBgkRpoAHAAeACAAe0CiAHADpIBBTItMy4zmAEAoAEBwAEB&sclient=gws-wiz-local#rlfi=hd:;si:;mv:[[17.454595986929753,78.36446229269447],[17.454152050801806,78.36377832939567

Kuttler, K. (2023, March 27). *7.1: Eigenvalues and eigenvectors of a matrix*. Mathematics LibreTexts. Retrieved April 6, 2023, from https://math.libretexts.org/Bookshelves/Linear_Algebra/A_First_Course_in_Linear_Algebra_(Kuttler)/07%3A_Spectral_Theory/7.01%3A_Eigenvalues_and_Eigenvectors_of_a_Matrix

Suhov, Y. (2009). *Markov chains 7. convergence to equilibrium. long-run pro- portions*. Statistical Laboratory, University of Cambridge. Retrieved April 6, 2023, from http://www.statslab.cam.ac.uk/~yms/M7_2.pdf