Year 8 Cat Assignment 2 – Markov Chains

"Amnesia: Partial or total memory loss"

In this assignment, you will investigate how processes which can be characterised as having amnesia(being memoryless), can be modelled using something known as a Markov Chain. This investigation will utilise concepts and knowledge from your study of probability theory. It will also extend you by introducing the concept of a matrix and the operation of matrix multiplication.

$$\begin{bmatrix} 2 & -3 & 0 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 5 \\ -3 \end{bmatrix}$$
$$= (2)(2) + (-3)(4) + (0)(5) + (4)(-3)$$
$$= -20.$$

1. Data structures

A data structure is a way data is stored and managed in computers, an example you should be familiar with is the list. You may have also encountered the dictionary. All data you access on a computer will be stored in some sort of data structure, including the letters in this word document and the pixel values that are colouring your screen.

Dictionaries allow for values to be matched with a key for storage in a dictionary. If you know the key, you can instantly access the data value. It's like how, if you know the first letter of a word, you can efficiently access the word and its definition in a dictionary book. Lists allow for pieces of data to be combined so they can be searched through one by one. They are not as quick and efficient at retrieval as Dictionaries are, but they have other strengths such as they preserve order.

You will need to be confident using dictionaries or lists or both in this assignment, so this section focused on ensuring this is the case.

Open *"DictionarySample.py"* to view a brief explainer of how Dictionaries work. Run the code and make sure you understand what each line is doing. Open *"ListSample.py"* to view a brief explainer of how Lists work. Run the code and make sure you understand what each line is doing.

You may need to refer to the sample code for the following tasks.

For the following sections, 1.1 to 1.4 inclusive, where you are required to write code, you may either write a separate program for each section, or just build a single program with a function that addresses each section.

1.1

The text file *"data_structure_data.txt"* contains many letters, all on a separate line. Write a program which reads in this file and prints on a single line each letter and the count of the amount of times the letter appears in the text file.

If you would like to use a dictionary, you could assign each unique letter in the text file as a key in the dictionary. The value for each key would then be the number of times that letter occurs in the text file.

If you would like to use a list, you could make a list as such [0,0,0,0,0] and then for every A you read in, add 1 to index 0, for every B you read in add 1 to index 1, and so on.

At the end your program should print something similar to:

D:2, E:4, A:2, C:4, B:10

- Save this program and include a screenshot of the code in your slides. Also include either a brief text description of how the algorithm works or a flow diagram. **(2 marks)**

1.2

Copy your previous program and rename the copy appropriately. Edit this program so that the value for each letter is not the count, but instead the number of times that letter appears divided by the total number of letters in the text file. At the end your program should print something like:

D:0.2, E:0.2, A:0.4, C:0.1, B:0.1

- Save this program and include a screenshot of the code in your slides. Also include either a brief text description of how the algorithm works or a flow diagram. **(4 marks)**

1.3

Open the following two programs and try to deduce what is going on:

- "DictionaryMultiple.py"
- "ListMultiple.py"

There is a lot going on in only a few lines so do not feel the need to understand it after a single review. Feel free to change the numbers in the dictionaries and to repeatedly calculate what you think will be printed.

- In your slides briefly describe what you think is going on in one of, or both of the programs (No more than 3 sentences). **(2 marks)**

1.4

Write a program that will read in the contents of "string_split.txt" and split each line around the delimiting character, which is in this case "*" (each line will start with either 1,2 or 3 then have a * then finish with 1,2 or 3). Then load into a dictionary or list the contents so that for each letter, you list the letters that follow it after the delimiter, along with the percentage of times that letter follows. You must use either a dictionary or list for this task.

For the following text file contents:

1*3

3*1

1*2

2*1

1*3

3*1

1*3

3*1

You would print the following if you used a dictionary:

1:{1:0,2:0.25,3:0.75},

2: {1:1,2:0,3:0},

3: {1:1,2:0,3:0}

- Save this program and include a screenshot of the code in your slides. Also include either a brief text description of how the algorithm works or a flow diagram. **(5 marks)**

Ensure the code you wrote to complete these sections is included in your submission.

2 Tree Diagrams

2.1 Tree Diagrams

Imagine a town where there are only two states of weather, it's either Sunny, or Rainy. Furthermore, each day's weather only depends upon the previous day. The meteorologist of the town has accurately determined the following rules:

- If the day is Sunny, there's an 80% chance it will be Sunny tomorrow, and a 20% chance it will be rainy.
- If the day is Rainy, there's an 40% chance it will be Sunny tomorrow, and a 60% chance it will be rainy.

If today is Sunny, represent this situation with a tree diagram that shows the possible probabilities for the weather of the next two days are. List the possible weather outcomes for the next two days and their associated probabilities.

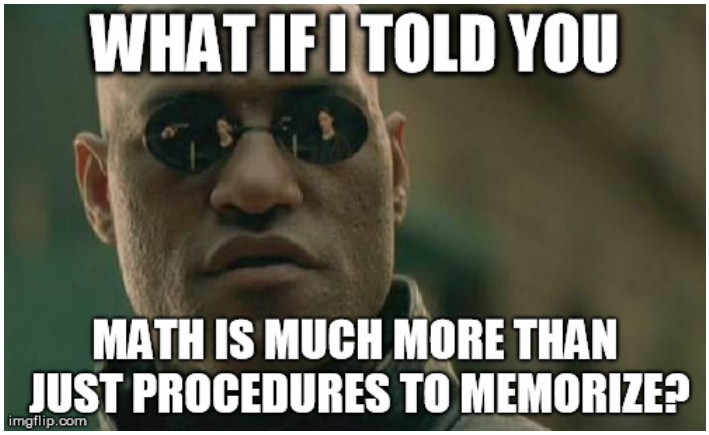
- Include this tree diagram in your slides. **(3 marks)**

2.2 Tree Diagrams

Following on from the previous scenario, you could also use a tree diagram to show the possible combinations of weather for the next 3 days. You could also use it for the next 4 days, or 10 days. Why would you not want to do this though? What might be a problem with using a tree diagram to model the next 10 days of weather or more for this town? How many outcomes would such a tree diagram have?

- Include a slide in your presentation that answers the above questions. **(2 marks)**

3. Introduction to Matrix Multiplication



In this section of the assignment, you will learn what a matrix is and how to do basic matrix multiplication. You will need this knowledge later in the assignment.

In maths, a matrix is a rectangular array(table) of numbers. Each number in the matrix is referred to as an element or entry. Matrices (plural of matrix) are very useful as they allow us to simultaneously multiply or add big groups of numbers in one go. If we look at the matrix below we can see it has 2 rows and 2 columns. It is referred to as a 2x2 matrix (two by two matrix).

$$\begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix}$$

Below is a matrix which has 1 row and 2 columns, we say it is a 1x2 matrix.

$$\begin{bmatrix} 0.1 & 0.9 \end{bmatrix}$$

Note that when we are listing the dimensions of the matrix (how many rows and columns it has) we list the row count first then the column count second after the 'x'. This ordering is important because it determines if we can multiply two matrices or not.

The rule is: if when we list the dimensions of the matrices we want to multiply, if the numbers next to each other are the same, we can multiply them, if not, we cannot. For example:

We can do the following matrix multiplication as the adjacent numbers of the dimensions are equal: 1x2 2x2.

$$\begin{bmatrix} 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix} = \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix} = [0.195\ 0.805]$$

Note how the dimensions of the new matrix are 1x2, it's like the middle two common dimensions cancelled out.

We can **not** do the following matrix multiplication as the adjacent numbers of the dimensions are equal: 2x2 1x2.

$$\begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix} [0.1 \quad 0.9] = Undefined$$

How does the multiplication actually work? You simply multiply each value of a row of the first matrix by the corresponding value of a column in the second matrix and then add them. The diagram below shows this process, which will happen repeatedly to get the answer on the right.

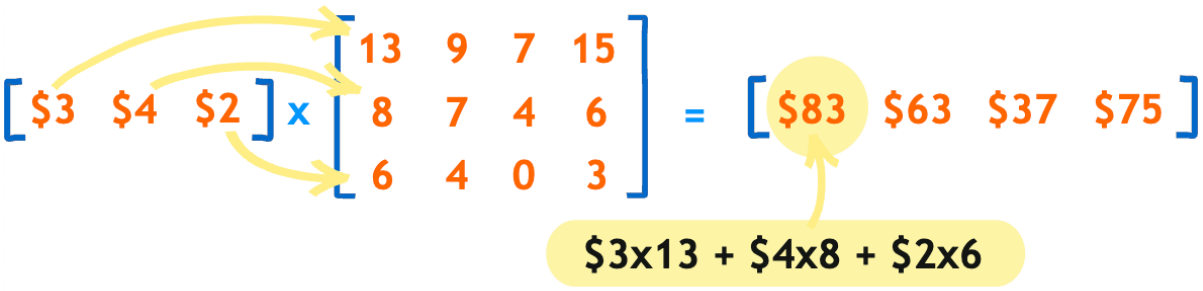


Figure 1
Matrix multiplication example.

3.1 Matrix Multiplication Questions

Example:

Take each value in the row matrix, and multiply it by the corresponding value in each column as is done below:

$$[0.1 \quad 0.9]\begin{bmatrix}0.6 & 0.4\\ 0.15 & 0.85\end{bmatrix} = [0.1 \times 0.6 + 0.9 \times 0.15 \quad 0.1 \times 0.4 + 0.9 \times 0.95]$$

$$[0.195 \quad 0.805]$$

Multiply the matrices below and then check your answers on the next page:

$$Q1)\ [9 \quad 4]\begin{bmatrix}1 & 2\\ 3 & 1\end{bmatrix} =$$

$$Q2)\ [6 \quad 2]\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} =$$

$$Q3)\ [0.1 \quad 0.4 \quad 0.5]\begin{bmatrix}0.2 & 0.3 & 0.5\\ 0.7 & 0.2 & 0.1\\ 0.6 & 0.2 & 0.2\end{bmatrix} =$$

$$Q4)\ [0.1 \quad 0.2 \quad 0.3 \quad 0.15 \quad 0.25]\begin{bmatrix}0.4 & 0.1 & 0.1 & 0.2 & 0.2\\ 0.1 & 0 & 0 & 0 & 0.9\\ 0.15 & 0.05 & 0.2 & 0.4 & 0.2\\ 0.3 & 0.1 & 0.3 & 0 & 0.3\\ 0.5 & 0 & 0.5 & 0 & 0\end{bmatrix} =$$

- Include pictures of handwritten worked solutions to these problems in your slides. **(5 marks)**

Solutions:

$$[9 \quad 4]\begin{bmatrix}1 & 2\\ 3 & 1\end{bmatrix} = [21 \quad 22]$$

$$[6 \quad 2]\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} = [6 \quad 2]$$

$$[0.1 \quad 0.4 \quad 0.5]\begin{bmatrix}0.2 & 0.3 & 0.5\\ 0.7 & 0.2 & 0.1\\ 0.6 & 0.2 & 0.2\end{bmatrix} = [0.6 \quad 0.21 \quad 0.19]$$

$$[0.1 \quad 0.2 \quad 0.3 \quad 0.15 \quad 0.25]\begin{bmatrix}0.4 & 0.1 & 0.1 & 0.2 & 0.2\\ 0.1 & 0 & 0 & 0 & 0.9\\ 0.15 & 0.05 & 0.2 & 0.4 & 0.2\\ 0.3 & 0.1 & 0.3 & 0 & 0.3\\ 0.5 & 0 & 0.5 & 0 & 0\end{bmatrix}$$
$$= [0.275 \quad 0.04 \quad 0.24 \quad 0.14 \quad 0.305]$$

3.2 Matrix Multiplication in Code

Now go and look at *"DictionaryMultiply.py"* and *"ListMultiply.py"*, edit them so they match what is below and run it and check the output. What do you notice?

"DictionaryMultiply.py"

```
state = {"a":9,"b":4}

matrix = {

  "a":{"a":1,"b":2},

  "b":{"a":3,"b":1},

     }
```

"ListMultiply.py"

```
state2 = [9,4]

matrix2 = [

  [1,2],

  [3,1]

]
```

The two files, *"DictionaryMultiply.py"* and *"ListMultiply.py"* have been given to you as examples of how you can do matrix multiplication in your code. You will need to perform this operation for the Markov Chain component of the assignment.

- Try to edit one of these files so that it does matrix multiplication for a matrix of a different size. For example, you may like to attempt to automatically solve Q3 or Q4. Save this code file and include a screenshot in your PowerPoint with a couple of comments explaining what changes you made. **(2 marks)**

4. Building a Markov Chain

In section 2, you considered a scenario where the possible future weather states of a town could be modelled with a tree diagram. The problem description stated that *"...Furthermore, each day's weather only depends upon the previous day..."* This comment means that the process we are modelling is ***memoryless***. The weather for tomorrow only depends on the weather today.

This memoryless feature of this situation means we can describe this situation as having the Markov Property. This also means we can model this situation with a Markov Chain as is done below. The circle with an S inside represents the town being in a Sunny State, and the circle with an R inside represents the town being in a Rainy state.

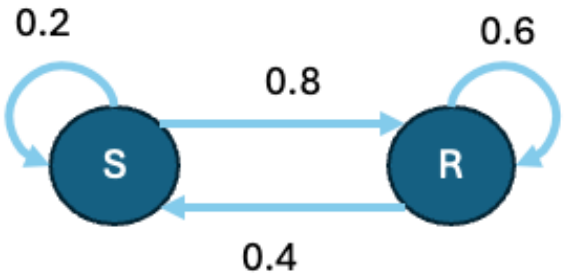


Figure 2

The arrows between the two states are referred to as transitions. When modelling the weather of this town, we can think about today's weather as moving along the arrows between the two states. For example, if today was Sunny, and then tomorrow was Rainy, we would move from S along the arrow towards R. Note that sum of the values assigned to each arrow leaving a single node all sum to 1.0. This is because these are probability values.

The transitions and their associated probabilities can be represented in what is known as a transition matrix, as shown below (Note each row sums to 1.0). Can you see how the diagram above and the table below tell us the same thing?

		To	
		Sunny	Rainy
From	Sunny	0.2	0.8
	Rainy	0.4	0.6

A question you might ask is, where did the values for the transitions come from? In reality they would likely be based on historical records. For example, the weather of the town may be recorded to show what the weather on subsequent days was recorded in the following manner:

S->R

R->R

R->S

S->R

R->S

S->S

S->R

R->R

R->R

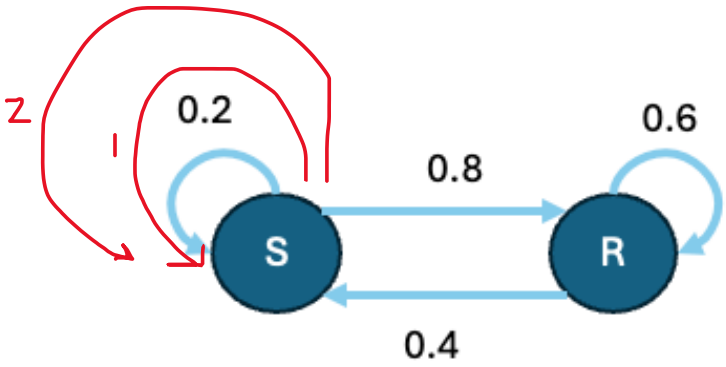
... and so on ...

Note how the end state for each line is the start state for the next line.

The transition probabilities could be estimated from this text file. To process such data to generate the probabilities for the transitions, you could consider using some of the techniques in section “1.DataStructures”.

Working out the State after a Set Number of Days

If we were asked, if today is Sunny, what is the probability that the next two days will be Sunny? We would follow the arrow which leaves S (our initial state) and returns into itself, twice.



This corresponds to the following calculation.

$$0.2 \times 0.2 = 0.04$$

There is a 4% chance that the next two days are Sunny, if today is Sunny. What is the probability that the next two days will be Rainy if today is Sunny?

Often the transition matrix is merely represented as a matrix, as is shown below:

$$\begin{bmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \end{bmatrix}$$

Representing it in this form is effective as it allows matrix multiplication to perform calculations quickly in parallel.

What if you wanted to determine the probability of all the possible combinations of weather for the next two days if today is Sunny?

Sunny(Initial State) then Sunny(Day 1) then Sunny(Day 2): 0.2x0.2=0.04
Sunny then Rainy then Sunny: 0.8x0.4=0.32
Sunny then Sunny then Rainy: 0.2x0.8=0.16
Sunny then Rainy then Rainy: 0.8x0.6=0.48

If today is Sunny, and we wanted to know the probability that the town would be Sunny after two days we would then just add 0.04 and 0.32 to get 0.36.

We can also answer the same question using Matrix multiplication.

Probabilities after 1 day:

$$[1.0 \quad 0.0]\begin{bmatrix}0.2 & 0.8\\ 0.4 & 0.6\end{bmatrix} = [1 \times 0.2 + 0.0 \times 0.4 \quad 0.1 \times 0.8 + 0.0 \times 0.6]$$

$$[0.1 \times 0.2 \quad 0.1 \times 0.8] = [0.2\ 0.8]$$

Probabilities after 2 days:

$$[0.2 \quad 0.8]\begin{bmatrix}0.2 & 0.8\\ 0.4 & 0.6\end{bmatrix} = [0.2 \times 0.2 + 0.8 \times 0.4 \quad 0.2 \times 0.8 + 0.8 \times 0.6]$$

$$[0.04 + 0.32 \quad 0.16 + 0.48] = [0.36\ 0.64]$$

The first entry tells us, that there is a 36% chance the weather will be sunny on the second day, and the second entry tells us that there is a 64% chance the weather will be sunny on the second day.

Can you see how Matrix multiplication works the same thing out?

To see the above matrix multiplications in code, look at “TwoDaysAfterSunny.py”.

If we were asked the question, what is the probability that a system will be in a particular state after 3 days, and we assume that system is memoryless, we can use a Markov chain to model it and we will setup our initial state matrix and multiply it by the transition matrix 3 times. Look at “ThreeDaysAfterRainyOrSunny.py” to see an example of this.

Working out the Equilibrium

We could repeat this procedure to tell us the probability the weather will be sunny or rainy after 10 days. Open “TenDaysAfterSunny.py” and run the code. What do you notice?

Now, change the initial state to something else, e.g.: [0.5,0.5] and run it again. What did you notice? The reason you got the same result is that [0.33,0.66] is the steady state of this Markov chain. It is the pair of probabilities that the system “converges” to after enough matrix multiplications are done. The values arrived at, will be arrived at no matter which starting state you use. This is known as the equilibrium state. In this assignment, anytime you work with a Markov Chain, that chain will converge to an equilibrium state.

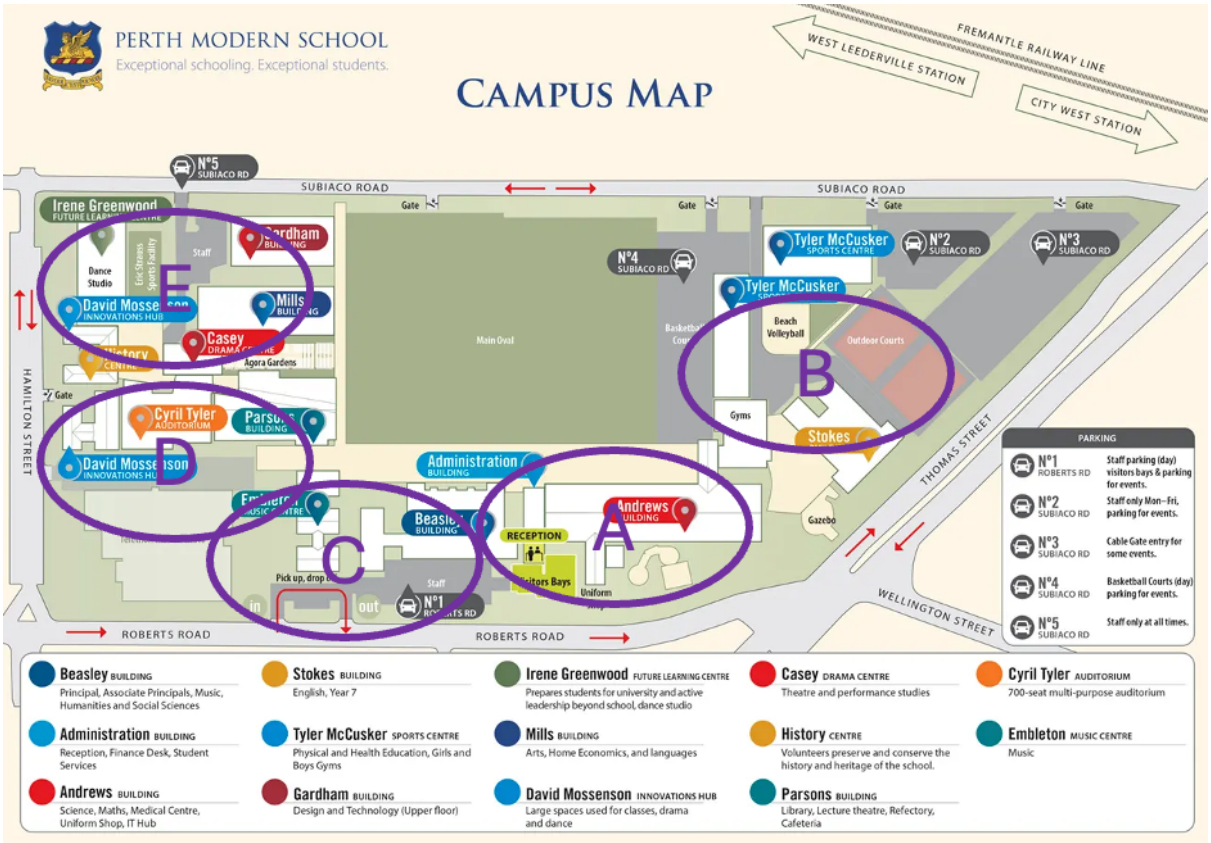
So, for our example, if we were asked in the long run will the town experience more Sunny or Rainy days, we could just multiply any initial state by the transition matrix a

large number of times and we will reach the equilibrium state. This will tell us no matter what the weather was on the first day you measured from, over a long period of time there will be a 33.33% chance it is sunny and a 66.66% chance it is rainy.

This could be useful information if the town council had a limited amount of resources to prepare the town for extreme weather. They could ask themselves, will the town spend more time in the Rainy State or Sunny State? Working out the equilibrium matrix would tell them the town is more often Rainy so spend the resources preparing for Rainy weather.

5. Applying Your Knowledge

Perth Modern's school network is served by 5 different routers all of which cover a section of the school. The routers are the devices that allow student computers to access services. For example, if you want to print at the school, your computer will speak to a router which will forward your request to a printer, if you want to go on the internet, your computer will speak to a router which will forward your request to a website. Routers also regularly communicate with each other to check how big the network its. When routers send information to one another, this information is called a packet.



Perth Modern's IT department wanted to analyse how the routers share traffic during the day, in short, they wanted to know which router works the hardest. To do this, they ran software overnight which released a single packet that bounced around the network from router to router, moving from each router to another, based on their normal forwarding preferences. When the routers forwarded the packet, they did not consider which router they received the packet from, as such this process is memoryless.

Logs of this router communication were recorded and saved in "network_traffic.txt".

The text file “network_traffic.txt” contains a log of the communications between a set of routers. A sample of the log is below:

A->B

B->C

C->E

E->A

A->D

D->B

B->D

D->E

The first line means, the packet transition from router A to router B. The second line means that the packet transitioned from B to C, and so on.

For this task you will build a program that models this data as a Markov chain.

Program Requirements

Program functionality – Your program should be able to do the following:

- Load in a text file formatted as “network_traffic.txt” and build an adjacency matrix from the file. **(5 marks)**
    - Note: See the program you built during section 1.4 to see an example of how to build an adjacency matrix.
- Allows the user to set the initial state. **(3 marks)**
- Outputs the calculated state after a custom amount of days. **(4 marks)**
    - Note: See “ThreeDaysAfterRainyOrSunny.py” to see an example of how to perform multiple matrix multiplications. It should be noted quite a lot of code repetition is happening here and there are other ways to achieve this matrix multiplication.
- Outputs to the user the equilibrium state of the system. **(3 marks)**

Program Design

- Include in a slide either a flow chart that demonstrates the functionality of your program or text that describes the role of each function and how they work together. **(8 marks)**

Menu Design

Design a menu that will allow the user to access all the functionality listed above under program requirements. **(5 marks)**

Code Quality

You will receive marks for code that is readable and has minimal unnecessary repetition. Using techniques such as loops or functions instead of just copying and pasting blocks of code will receive marks. Naming variables and functions in a common sense way so their use is clear will also receive marks. **(8 marks)**

Markov Chain Diagram

Include a Markov Chain diagram (similar to figure 2) for the text file “network_traffic.txt”. **(4 marks)**

Live Testing

Upon assignment completion you will complete a short test where your knowledge of your code and the project will be assessed. **(10 marks)**

Git

As with past assignments you are expected to use Git to track your progress. At the outset of your project, you should initialise a git repository in a folder named in the following form: FirstName_Surname_8_Cat_2

So John Smith's folder will be named as follows: John_Smith_8_Cat_2. You will initialise a git repository in this folder and it will be where you store all your assignment files. When working through this assignment, for sections that require you to produce code, you may decide whether you produce one big program that has different modules that complete sub section tasks, or whether you make a smaller python program for each sub section. However for section 5 you must produce a single program that has all the listed functionality.

You should regularly make commits as you progress through the assignment. Each commit should have a description of what progress you have made. Frequent and detailed commits are required to earn marks for this section. Consideration will also be given to use of comments and code readability. **(25 marks)**

Presentation

You will present your assignment to a small group as you have for past coding assignments. This presentation should go for roughly 12 minutes per person. Please have your code ready to be run for the presentation so you may demonstrate its effectiveness to your group.

Submission

Your assignment is due on the 24th of October, a submission link will be emailed out nearer to the date. Your submission folder should include all the code files you produced to complete each section along with your slides. You are not permitted to edit your presentation slides after you have submitted your project.

References

Math is Fun. (n.d.). *Matrix multiplication*. In *Math is Fun*. https://www.mathsisfun.com/algebra/matrix-multiplying.html