IME625A

Applications of Graph Theory in Markov Chains
&
Page Rank Algorithm

Group - F

Markov Chain

- A Markov Chain is a Stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained on the previous state.
- A stochastic process has the Markov property if the conditional probability distribution of future states of the process depends only upon the present state, not on the sequence of events that preceded it.

Markov State Diagram

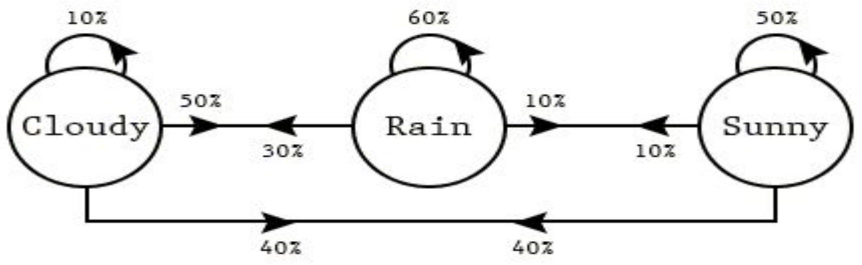


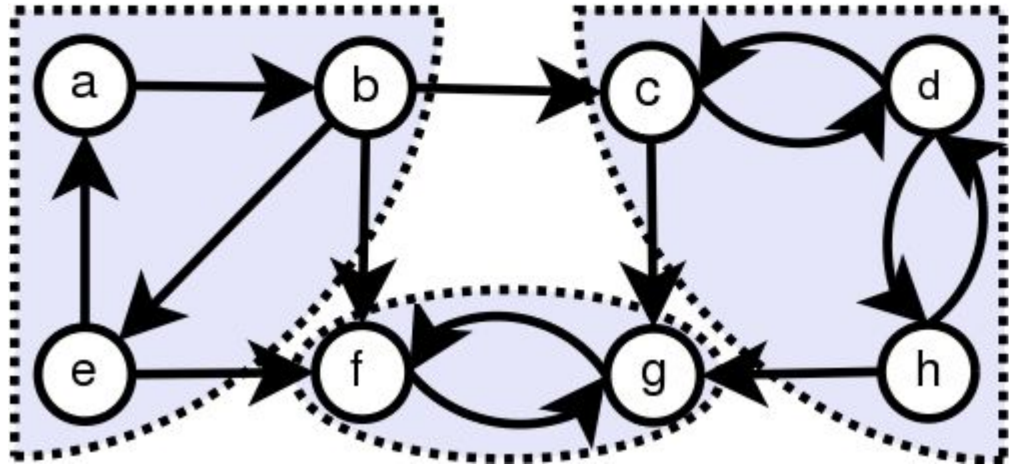
Figure 2

Reading the input

- We read transition matrix and starting distribution corresponding to markov chain.
- We then checks whether the given matrix is valid transition matrix or not.
- Basic constraints like
  - Each entry >= 0 and <= 1
  - Row sum equal to 1
- Validity check for starting distribution is also done.

Strongly Connected Component (SCC)

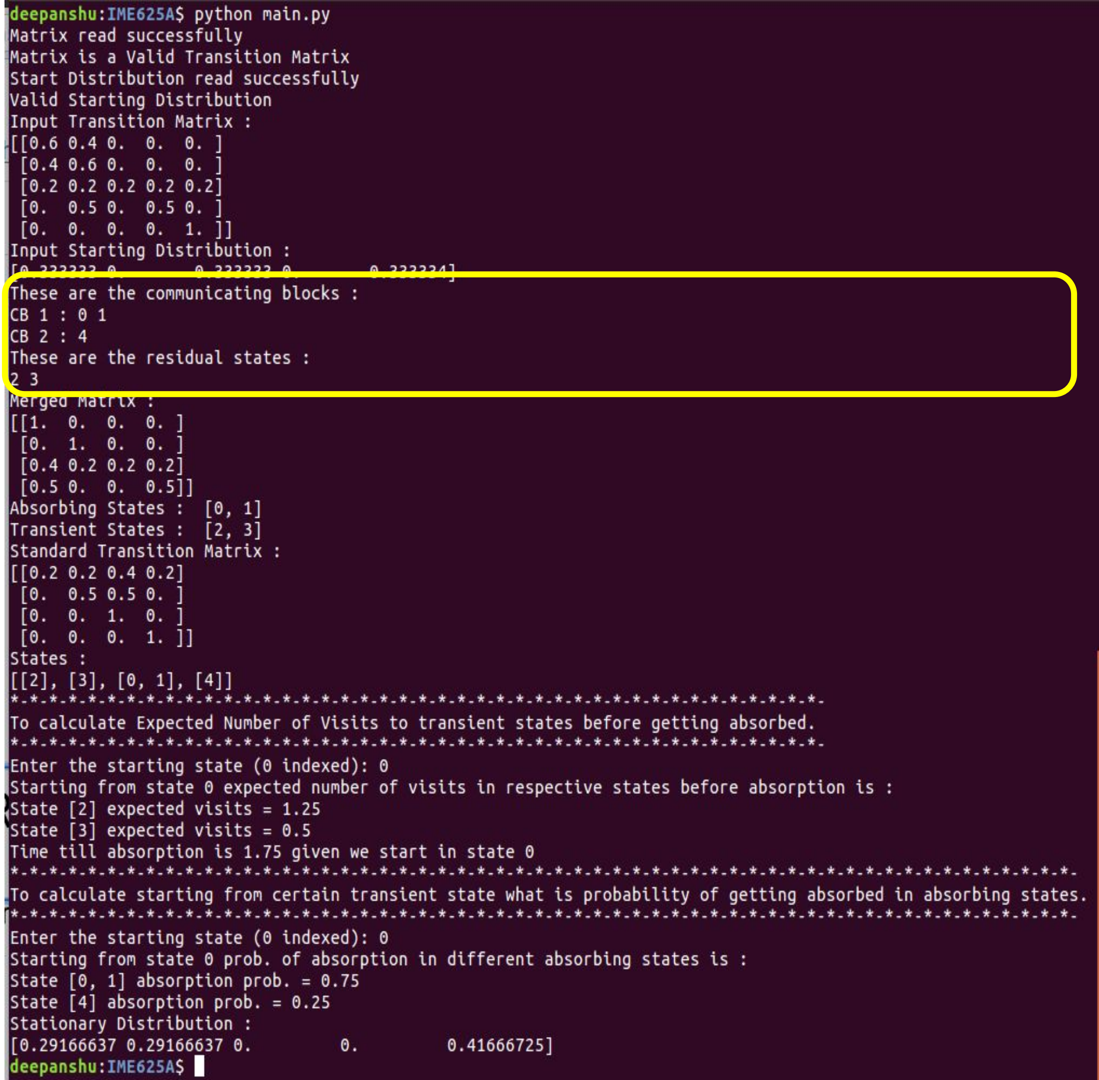
- A directed graph is strongly connected if there is a path between all pairs of vertices.
- The **strongly connected components** of an arbitrary directed graph form a partition into subgraphs that are themselves strongly connected.
- We used Kosaraju's algorithm to compute SCC.



To calculate ...

- After finding the Scc we need to classify if it is Communicating Block or Residual State.
- To check that we need to see If there is at least one vertex in the SCC which has an edge out.
- If there isn’t then it’s a communicating block else residual state.

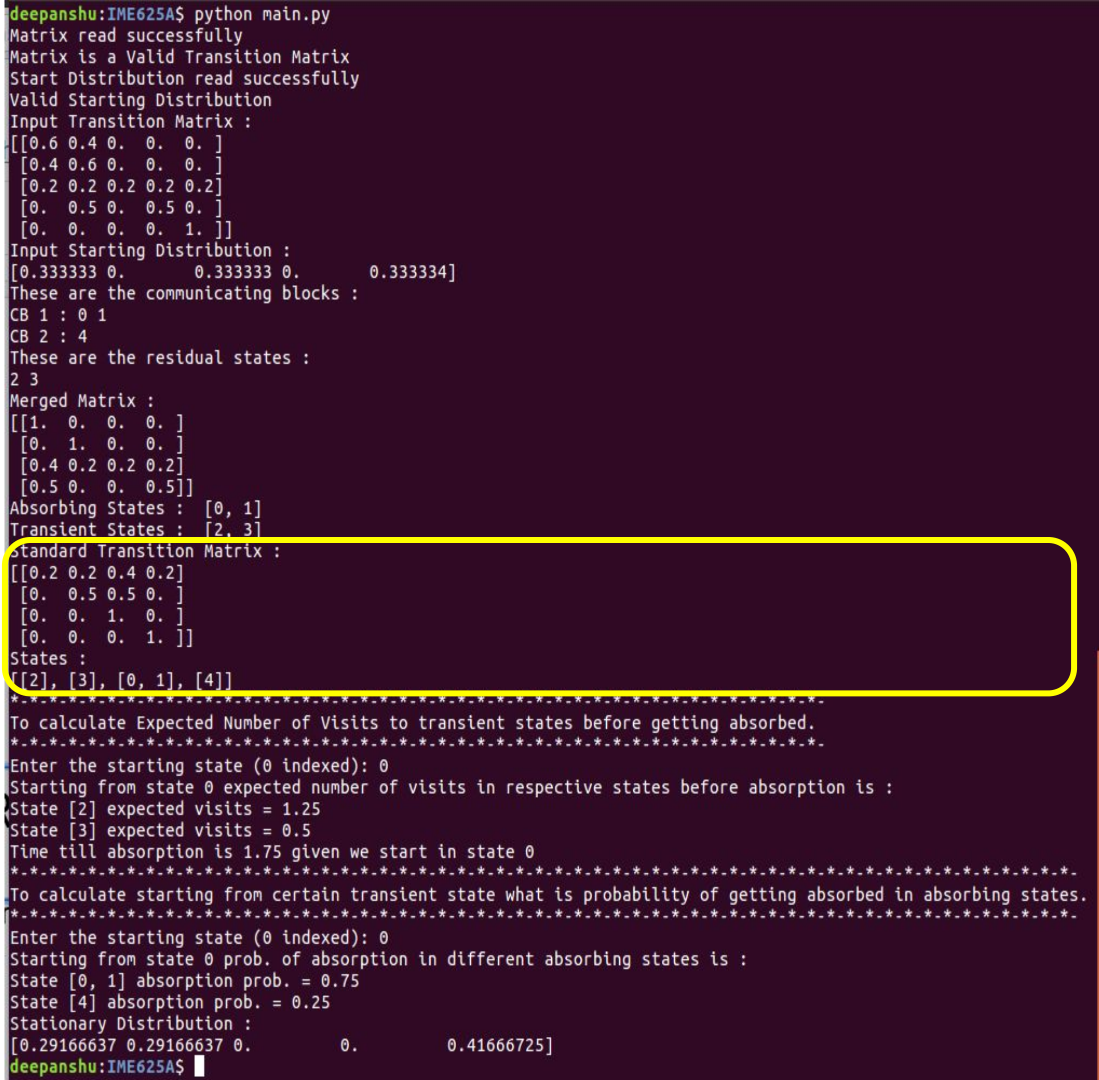
Communicating Blocks



```
deepanshu:IME625A$ python main.py
Matrix read successfully
Matrix is a Valid Transition Matrix
Start Distribution read successfully
Valid Starting Distribution
Input Transition Matrix :
[[0.6 0.4 0.  0.  0. ]
 [0.4 0.6 0.  0.  0. ]
 [0.2 0.2 0.2 0.2 0.2]
 [0.  0.5 0.  0.5 0. ]
 [0.  0.  0.  0.  1. ]]
Input Starting Distribution :
[0.333333 0.       0.333333 0.       0.333334]
These are the communicating blocks :
CB 1 : 0 1
CB 2 : 4
These are the residual states :
2 3
Merged Matrix :
[[1.  0.  0.  0. ]
 [0.  1.  0.  0. ]
 [0.4 0.2 0.2 0.2]
 [0.5 0.  0.  0.5]]
Absorbing States :  [0, 1]
Transient States :  [2, 3]
Standard Transition Matrix :
[[0.2 0.2 0.4 0.2]
 [0.  0.5 0.5 0. ]
 [0.  0.  1.  0. ]
 [0.  0.  0.  1. ]]
States :
[[2], [3], [0, 1], [4]]
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
To calculate Expected Number of Visits to transient states before getting absorbed.
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
Enter the starting state (0 indexed): 0
Starting from state 0 expected number of visits in respective states before absorption is :
State [2] expected visits = 1.25
State [3] expected visits = 0.5
Time till absorption is 1.75 given we start in state 0
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
To calculate starting from certain transient state what is probability of getting absorbed in absorbing states.
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
Enter the starting state (0 indexed): 0
Starting from state 0 prob. of absorption in different absorbing states is :
State [0, 1] absorption prob. = 0.75
State [4] absorption prob. = 0.25
Stationary Distribution :
[0.29166637 0.29166637 0.         0.         0.41666725]
deepanshu:IME625A$
```

Canonical Form

$$P = \begin{array}{c|c|c} & \text{TR.} & \text{ABS.} \\ \hline \text{TR.} & Q & R \\ \hline \text{ABS.} & 0 & I \end{array}$$

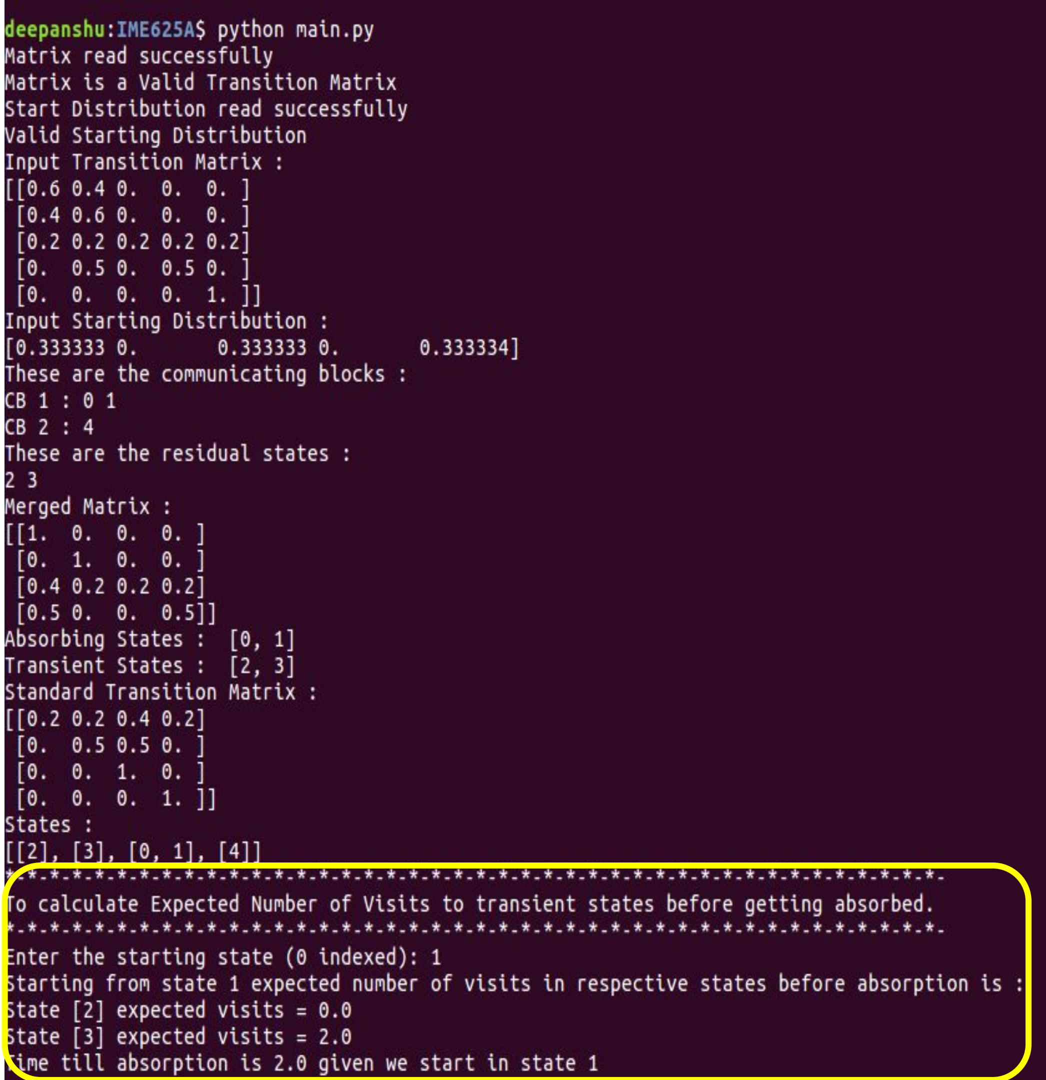


```
deepanshu:IME625A$ python main.py
Matrix read successfully
Matrix is a Valid Transition Matrix
Start Distribution read successfully
Valid Starting Distribution
Input Transition Matrix :
[[0.6 0.4 0.  0.  0. ]
 [0.4 0.6 0.  0.  0. ]
 [0.2 0.2 0.2 0.2 0.2]
 [0.  0.5 0.  0.5 0. ]
 [0.  0.  0.  0.  1. ]]
Input Starting Distribution :
[0.333333 0.       0.333333 0.       0.333334]
These are the communicating blocks :
CB 1 : 0 1
CB 2 : 4
These are the residual states :
2 3
Merged Matrix :
[[1.  0.  0.  0. ]
 [0.  1.  0.  0. ]
 [0.4 0.2 0.2 0.2]
 [0.5 0.  0.  0.5]]
Absorbing States :  [0, 1]
Transient States :  [2, 3]
Standard Transition Matrix :
[[0.2 0.2 0.4 0.2]
 [0.  0.5 0.5 0. ]
 [0.  0.  1.  0. ]
 [0.  0.  0.  1. ]]
States :
[[2], [3], [0, 1], [4]]
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
To calculate Expected Number of Visits to transient states before getting absorbed.
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
Enter the starting state (0 indexed): 0
Starting from state 0 expected number of visits in respective states before absorption is :
State [2] expected visits = 1.25
State [3] expected visits = 0.5
Time till absorption is 1.75 given we start in state 0
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
To calculate starting from certain transient state what is probability of getting absorbed in absorbing states.
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
Enter the starting state (0 indexed): 0
Starting from state 0 prob. of absorption in different absorbing states is :
State [0, 1] absorption prob. = 0.75
State [4] absorption prob. = 0.25
Stationary Distribution :
[0.29166637 0.29166637 0.         0.         0.41666725]
deepanshu:IME625A$
```

Expected Num. of Visits

$$W = (I-Q)^{-1}$$

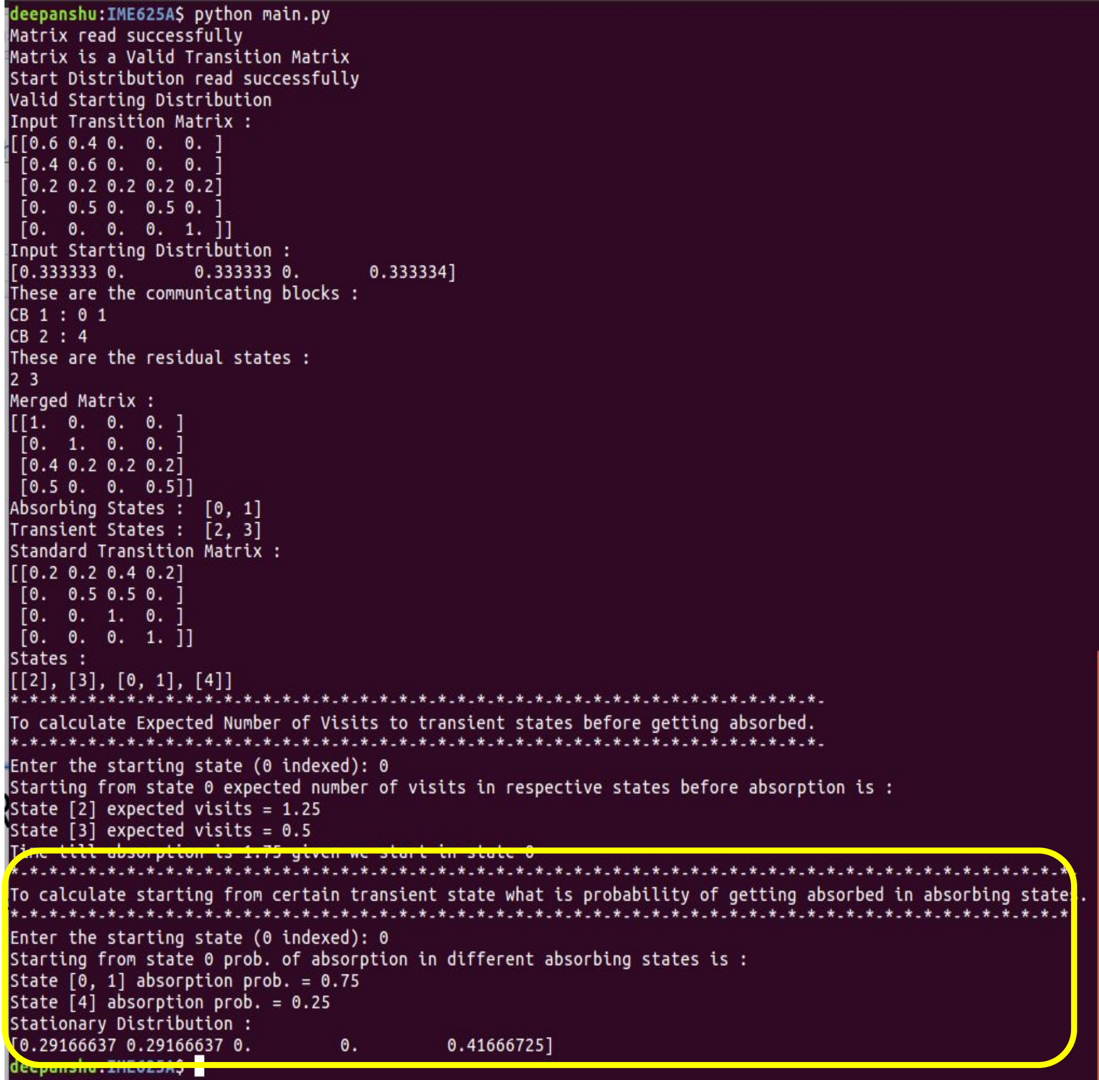
To calculate time till absorption we sum over the columns for a particular row ie. state



```
deepanshu:IME625A$ python main.py
Matrix read successfully
Matrix is a Valid Transition Matrix
Start Distribution read successfully
Valid Starting Distribution
Input Transition Matrix :
[[0.6 0.4 0.  0.  0. ]
 [0.4 0.6 0.  0.  0. ]
 [0.2 0.2 0.2 0.2 0.2]
 [0.  0.5 0.  0.5 0. ]
 [0.  0.  0.  0.  1. ]]
Input Starting Distribution :
[0.333333 0.       0.333333 0.       0.333334]
These are the communicating blocks :
CB 1 : 0 1
CB 2 : 4
These are the residual states :
2 3
Merged Matrix :
[[1.  0.  0.  0. ]
 [0.  1.  0.  0. ]
 [0.4 0.2 0.2 0.2]
 [0.5 0.  0.  0.5]]
Absorbing States :  [0, 1]
Transient States :  [2, 3]
Standard Transition Matrix :
[[0.2 0.2 0.4 0.2]
 [0.  0.5 0.5 0. ]
 [0.  0.  1.  0. ]
 [0.  0.  0.  1. ]]
States :
[[2], [3], [0, 1], [4]]
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
To calculate Expected Number of Visits to transient states before getting absorbed.
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
Enter the starting state (0 indexed): 1
Starting from state 1 expected number of visits in respective states before absorption is :
State [2] expected visits = 0.0
State [3] expected visits = 2.0
ime till absorption is 2.0 given we start in state 1
```

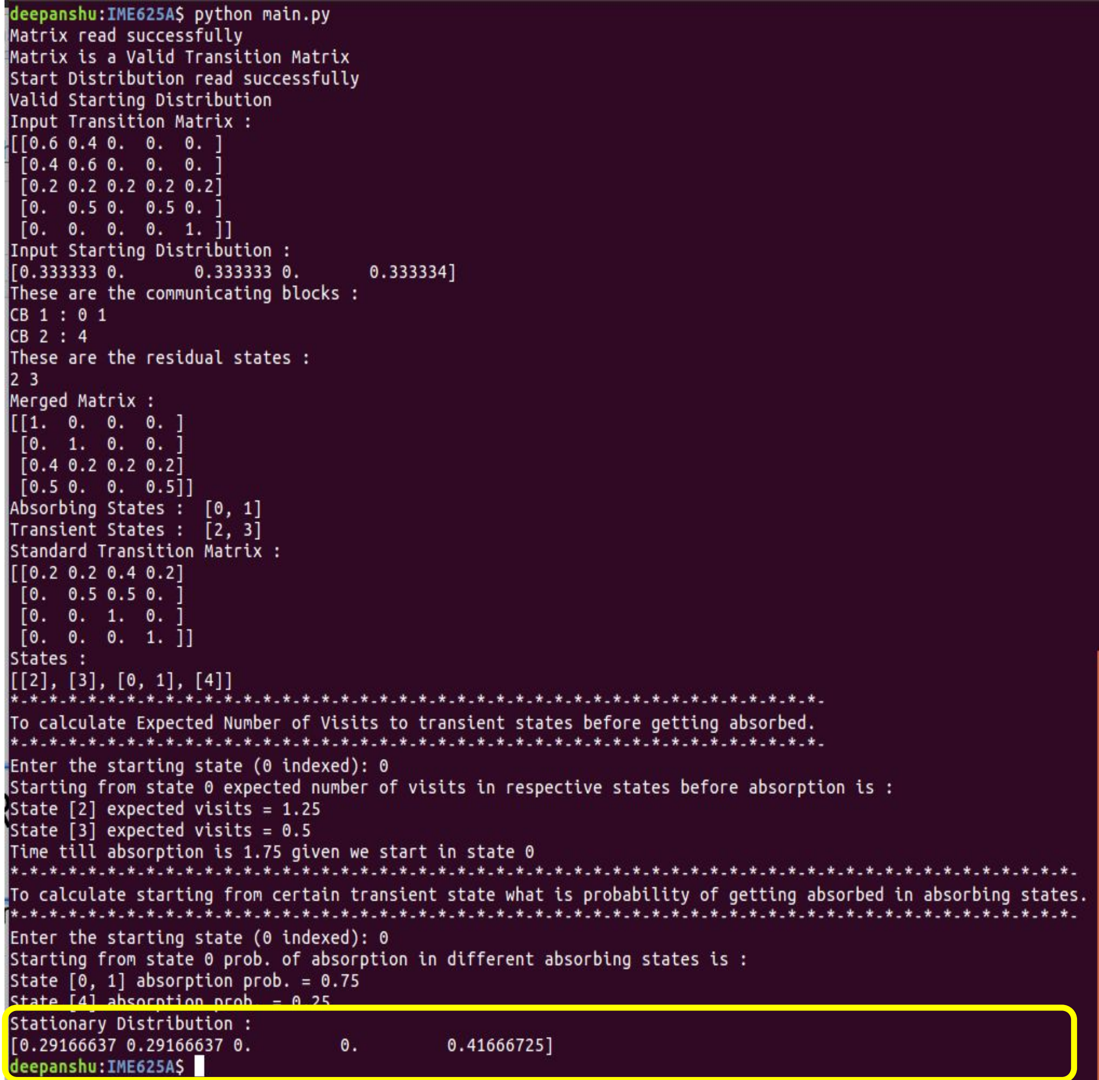
Hitting Probabilities

$$U = (I-Q)^{-1}R$$



```
deepanshu:IME625A$ python main.py
Matrix read successfully
Matrix is a Valid Transition Matrix
Start Distribution read successfully
Valid Starting Distribution
Input Transition Matrix :
[[0.6 0.4 0.  0.  0. ]
 [0.4 0.6 0.  0.  0. ]
 [0.2 0.2 0.2 0.2 0.2]
 [0.  0.5 0.  0.5 0. ]
 [0.  0.  0.  0.  1. ]]
Input Starting Distribution :
[0.333333 0.       0.333333 0.       0.333334]
These are the communicating blocks :
CB 1 : 0 1
CB 2 : 4
These are the residual states :
2 3
Merged Matrix :
[[1.  0.  0.  0. ]
 [0.  1.  0.  0. ]
 [0.4 0.2 0.2 0.2]
 [0.5 0.  0.  0.5]]
Absorbing States :  [0, 1]
Transient States :  [2, 3]
Standard Transition Matrix :
[[0.2 0.2 0.4 0.2]
 [0.  0.5 0.5 0. ]
 [0.  0.  1.  0. ]
 [0.  0.  0.  1. ]]
States :
[[2], [3], [0, 1], [4]]
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
To calculate Expected Number of Visits to transient states before getting absorbed.
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
Enter the starting state (0 indexed): 0
Starting from state 0 expected number of visits in respective states before absorption is :
State [2] expected visits = 1.25
State [3] expected visits = 0.5
Time till absorption is 1.75 given we start in state 0
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
To calculate starting from certain transient state what is probability of getting absorbed in absorbing states.
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
Enter the starting state (0 indexed): 0
Starting from state 0 prob. of absorption in different absorbing states is :
State [0, 1] absorption prob. = 0.75
State [4] absorption prob. = 0.25
Stationary Distribution :
[0.29166637 0.29166637 0.         0.         0.41666725]
deepanshu:IME625A$
```

Stationary Distribution



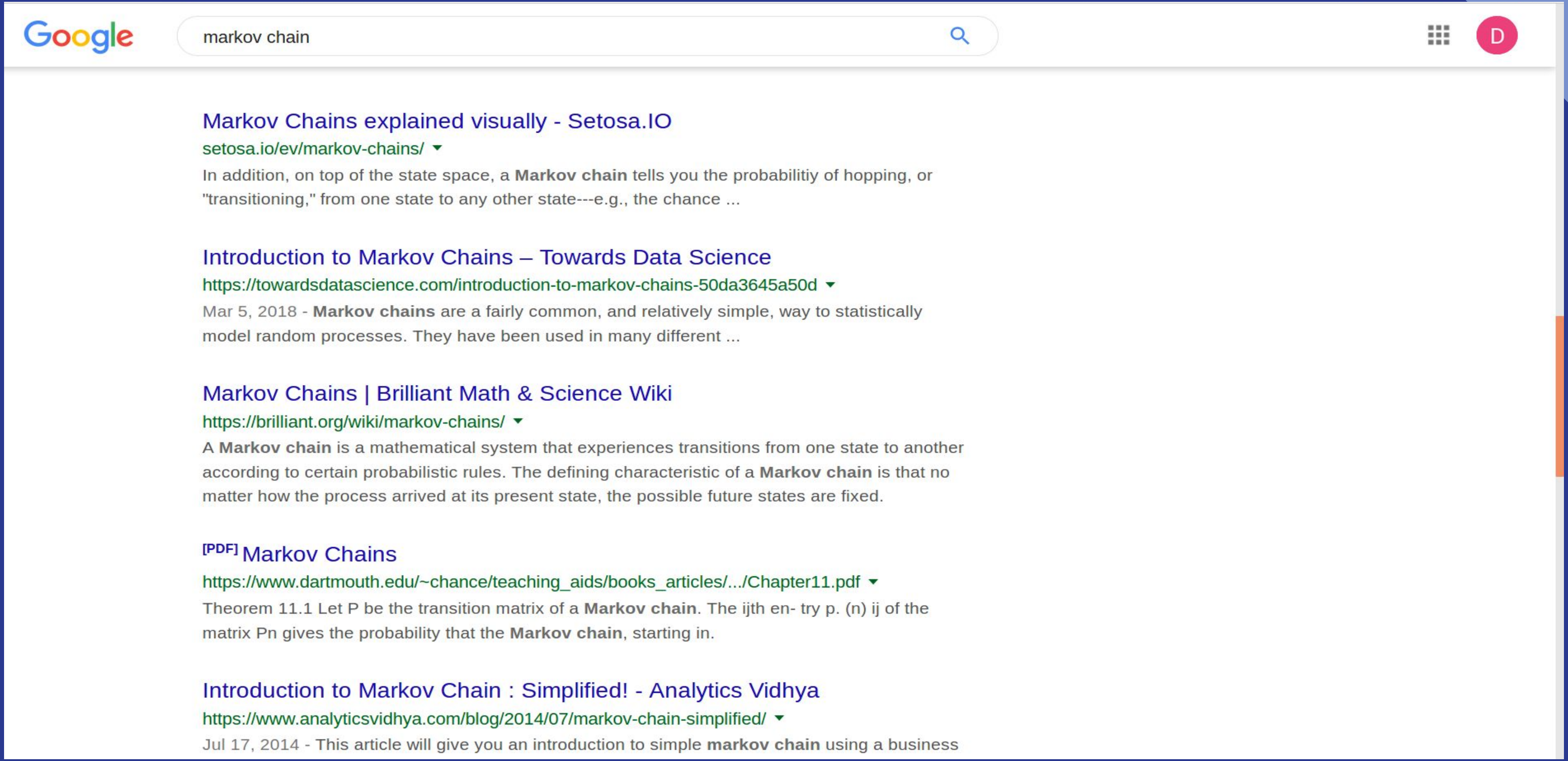
```
deepanshu:IME625A$ python main.py
Matrix read successfully
Matrix is a Valid Transition Matrix
Start Distribution read successfully
Valid Starting Distribution
Input Transition Matrix :
[[0.6 0.4 0.  0.  0. ]
 [0.4 0.6 0.  0.  0. ]
 [0.2 0.2 0.2 0.2 0.2]
 [0.  0.5 0.  0.5 0. ]
 [0.  0.  0.  0.  1. ]]
Input Starting Distribution :
[0.333333 0.       0.333333 0.       0.333334]
These are the communicating blocks :
CB 1 : 0 1
CB 2 : 4
These are the residual states :
2 3
Merged Matrix :
[[1.  0.  0.  0. ]
 [0.  1.  0.  0. ]
 [0.4 0.2 0.2 0.2]
 [0.5 0.  0.  0.5]]
Absorbing States :  [0, 1]
Transient States :  [2, 3]
Standard Transition Matrix :
[[0.2 0.2 0.4 0.2]
 [0.  0.5 0.5 0. ]
 [0.  0.  1.  0. ]
 [0.  0.  0.  1. ]]
States :
[[2], [3], [0, 1], [4]]
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
To calculate Expected Number of Visits to transient states before getting absorbed.
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
Enter the starting state (0 indexed): 0
Starting from state 0 expected number of visits in respective states before absorption is :
State [2] expected visits = 1.25
State [3] expected visits = 0.5
Time till absorption is 1.75 given we start in state 0
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
To calculate starting from certain transient state what is probability of getting absorbed in absorbing states.
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.
Enter the starting state (0 indexed): 0
Starting from state 0 prob. of absorption in different absorbing states is :
State [0, 1] absorption prob. = 0.75
State [4] absorption prob. = 0.25
Stationary Distribution :
[0.29166637 0.29166637 0.         0.         0.41666725]
deepanshu:IME625A$
```

PageRank Algorithm

- PageRank is an algorithm used by Google Search to rank websites in their search engine results
- PageRank is a way of measuring the importance of website pages
- According to Google

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

Sample Web Query



markov chain

D

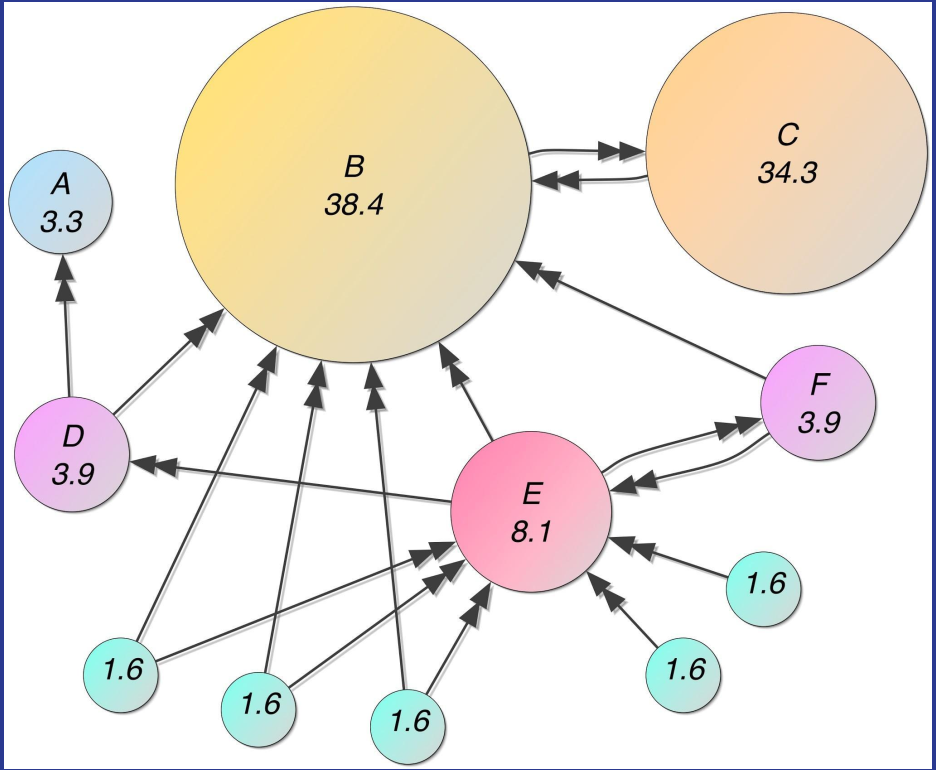
Markov Chains explained visually - Setosa.IO
setosa.io/ev/markov-chains/
In addition, on top of the state space, a **Markov chain** tells you the probabilitiy of hopping, or "transitioning," from one state to any other state---e.g., the chance ...

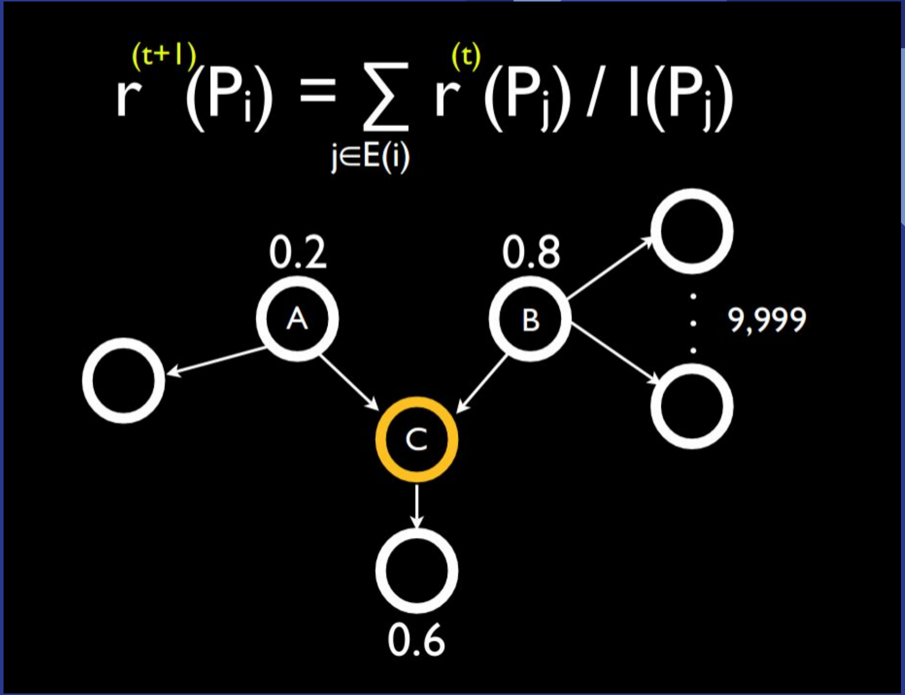
Introduction to Markov Chains – Towards Data Science
https://towardsdatascience.com/introduction-to-markov-chains-50da3645a50d
Mar 5, 2018 - **Markov chains** are a fairly common, and relatively simple, way to statistically model random processes. They have been used in many different ...

Markov Chains | Brilliant Math & Science Wiki
https://brilliant.org/wiki/markov-chains/
A **Markov chain** is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules. The defining characteristic of a **Markov chain** is that no matter how the process arrived at its present state, the possible future states are fixed.

[PDF] Markov Chains
https://www.dartmouth.edu/~chance/teaching_aids/books_articles/.../Chapter11.pdf
Theorem 11.1 Let P be the transition matrix of a **Markov chain**. The ijth en- try p. (n) ij of the matrix Pn gives the probability that the **Markov chain**, starting in.

Introduction to Markov Chain : Simplified! - Analytics Vidhya
https://www.analyticsvidhya.com/blog/2014/07/markov-chain-simplified/
Jul 17, 2014 - This article will give you an introduction to simple **markov chain** using a business




$$r^{(t+1)}(P_i) = \sum_{j \in E(i)} r^{(t)}(P_j) / l(P_j)$$

Dataset

- We used webbot crawl hollins.edu dataset
- It became publicly available in Jan ‘04
- It consists of 6,012 webpages
- There are 23,875 connection between these webpages
- Source - https://www.limfinity.com/ir/

Our Model

- Each state corresponds to the webpage. There are 6012 states.
- We created 6012 x 6012 transition matrix $[a_{ij}]_{6012 \times 6012}$
- $a_{ij} =$ = 1, if there is a link to webpage j from webpage i, otherwise zero
- We then divided each entry by the corresponding row sum to make it a valid markov chain transition matrix

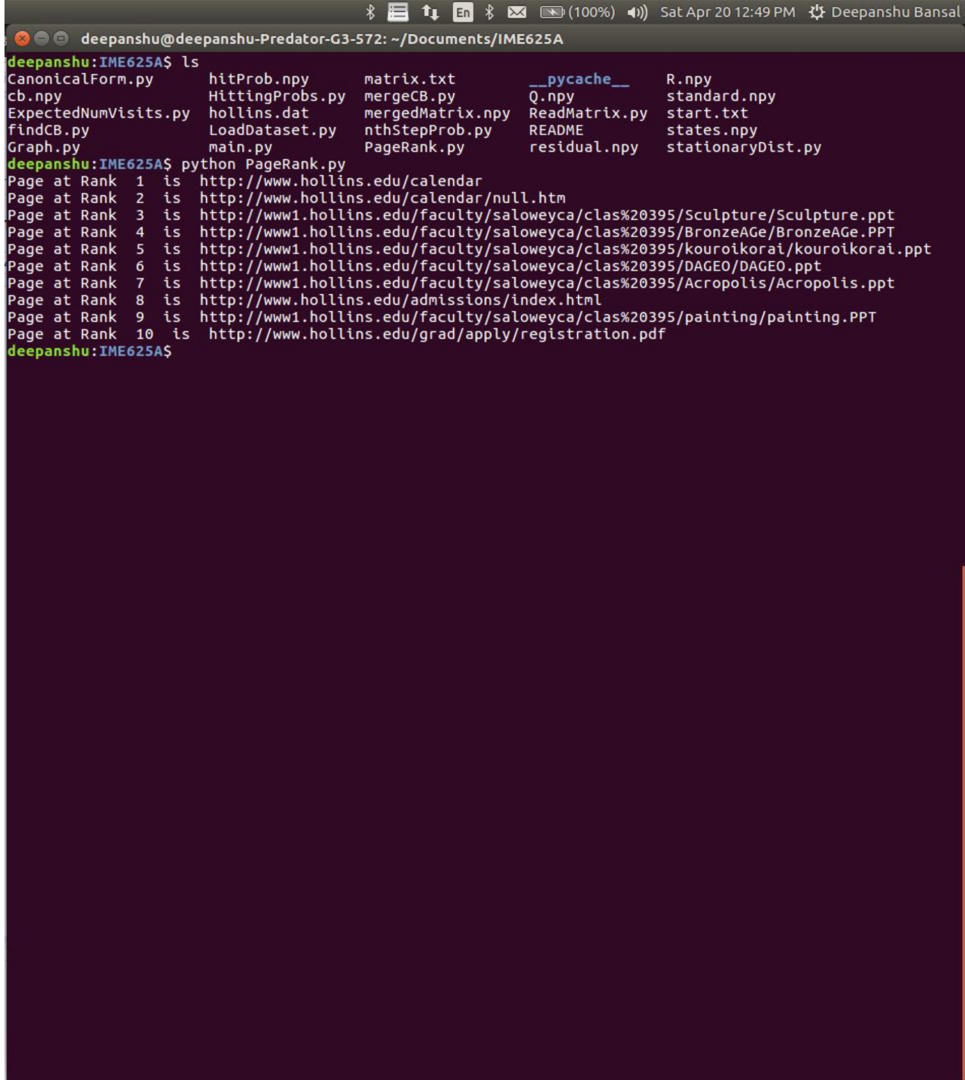
Results

- We calculated the stationary distribution of the markov chain
- We then sorted the different states according to their probabilities
- We output the top 10 web pages which will be visited mostly in the long run or in other words more relevant

We got ...

- Page at Rank 1 is http://www.hollins.edu/calendar
- Page at Rank 2 is http://www.hollins.edu/calendar/null.htm
- Page at Rank 3 is http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/Sculpture.ppt
- Page at Rank 4 is http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/BronzeAGe.PPT
- Page at Rank 5 is http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/kouroikorai.ppt
- Page at Rank 6 is http://www1.hollins.edu/faculty/saloweyca/clas%20395/DAGEO/DAGEO.ppt
- Page at Rank 7 is http://www1.hollins.edu/faculty/saloweyca/clas%20395/Acropolis/Acropolis.ppt
- Page at Rank 8 is http://www.hollins.edu/admissions/index.html
- Page at Rank 9 is http://www1.hollins.edu/faculty/saloweyca/clas%20395/painting/painting.PPT
- Page at Rank 10 is http://www.hollins.edu/grad/apply/registration.pdf

Demo



```
deepanshu:IME625A$ ls
CanonicalForm.py      hitProb.npy      matrix.txt         __pycache__    R.npy
cb.npy                HittingProbs.py  mergeCB.py         Q.npy          standard.npy
ExpectedNumVisits.py  hollins.dat      mergedMatrix.npy   ReadMatrix.py  start.txt
findCB.py             LoadDataset.py   nthStepProb.py     README         states.npy
Graph.py              main.py          PageRank.py        residual.npy   stationaryDist.py
deepanshu:IME625A$ python PageRank.py
Page at Rank  1  is  http://www.hollins.edu/calendar
Page at Rank  2  is  http://www.hollins.edu/calendar/null.htm
Page at Rank  3  is  http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/Sculpture.ppt
Page at Rank  4  is  http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/BronzeAGe.PPT
Page at Rank  5  is  http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/kouroikorai.ppt
Page at Rank  6  is  http://www1.hollins.edu/faculty/saloweyca/clas%20395/DAGEO/DAGEO.ppt
Page at Rank  7  is  http://www1.hollins.edu/faculty/saloweyca/clas%20395/Acropolis/Acropolis.ppt
Page at Rank  8  is  http://www.hollins.edu/admissions/index.html
Page at Rank  9  is  http://www1.hollins.edu/faculty/saloweyca/clas%20395/painting/painting.PPT
Page at Rank  10  is  http://www.hollins.edu/grad/apply/registration.pdf
deepanshu:IME625A$
```

The Team

150219
Deepanshu Bansal
150121
Anupriy
150571
Raushan Kumar

Thank You