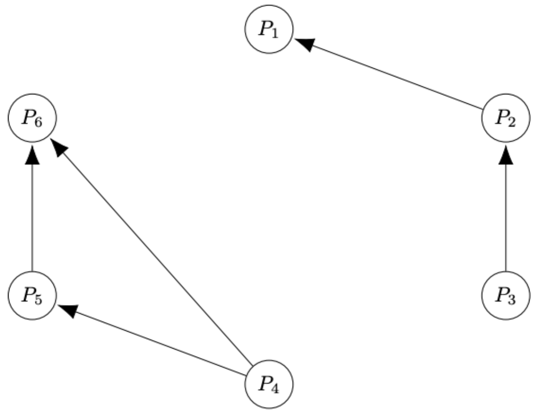
MATH401 Homework 8

James Zhang*

October 30, 2023

*Email: jzhang72@terpmail.umd.com

Exercise 7.4. The Google Pagerank algorithm works even if the internet is disconnected. Consider this example.



Find the pagerank of each of the pages.

Solution. Let us find our matrix T.

$$T = \frac{0.15}{6}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} + 0.85 \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}^T$$

$$T = \begin{bmatrix} 0.1667 & 0.875 & 0.025 & 0.025 & 0.025 & 0.1667 \\ 0.1667 & 0.025 & 0.875 & 0.025 & 0.025 & 0.1667 \\ 0.1667 & 0.025 & 0.025 & 0.025 & 0.025 & 0.1667 \\ 0.1667 & 0.025 & 0.025 & 0.025 & 0.025 & 0.1667 \\ 0.1667 & 0.025 & 0.025 & 0.45 & 0.025 & 0.1667 \\ 0.1667 & 0.025 & 0.025 & 0.45 & 0.875 & 0.1667 \end{bmatrix}$$

Note that T is a regular transition matrix so we can follow the theorem and obtain the eigenvector corresponding to the eigenvalue $\lambda = 1$ and then normalizing it such that the sum of the vector is 1. Doing this, we obtain

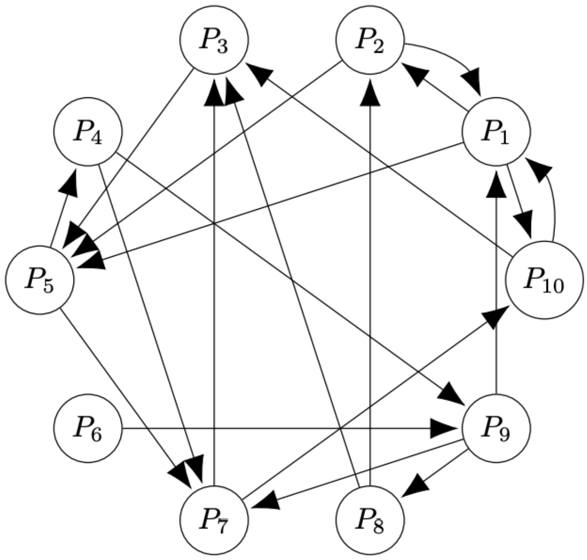
$$v_1 = \begin{bmatrix} 0.2454 & 0.1765 & 0.0954 & 0.0954 & 0.1359 & 0.2515 \end{bmatrix}^T$$

and so the pagerank is

- P_1 has pagerank 0.2454
- P_2 has pagerank 0.1765
- P_3 has pagerank 0.0954
- P_4 has pagerank 0.0954
- P_5 has pagerank 0.1359
- P_6 has pagerank 0.2515

□

Exercise 7.8. Find the pagerank of the pages in the following internet. You will definitely want to use technology for this!



Solution.

$$T = \frac{0.15}{10}\text{ones}(10,\ 10) + 0.85X$$

where X is the transition matrix such that the ij entry represents an outgoing edge from page j to page i. I have omitted it here because it is a large 10 by 10 matrix, but I use it in the MatLab calculations. Thus the resulting eigenvector is

```
T =

    0.0150    0.4400    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.2983    0.4400
    0.2983    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.4400    0.0150    0.0150
    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.4400    0.4400    0.0150    0.4400
    0.0150    0.0150    0.0150    0.0150    0.4400    0.0150    0.0150    0.0150    0.0150    0.0150
    0.2983    0.4400    0.8650    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150
    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150
    0.0150    0.0150    0.0150    0.4400    0.4400    0.0150    0.0150    0.0150    0.2983    0.0150
    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.0150    0.2983    0.0150
    0.0150    0.0150    0.0150    0.4400    0.0150    0.8650    0.0150    0.0150    0.0150    0.0150
    0.2983    0.0150    0.0150    0.0150    0.0150    0.0150    0.4400    0.0150    0.0150    0.0150
```

$$v_1 = \begin{bmatrix} 0.1086 & 0.0605 & 0.1457 & 0.0980 & 0.1953 & 0.0150 & 0.1593 & 0.0347 & 0.0694 & 0.1135 \end{bmatrix}$$

and so the resulting pagerank is

- P_1 has pagerank 0.1086
- P_2 has pagerank 0.0605
- P_3 has pagerank 0.1457
- P_4 has pagerank 0.0980
- P_5 has pagerank 0.1953

- P_6 has pagerank 0.0150
- P_7 has pagerank 0.1593
- P_8 has pagerank 0.0347
- P_9 has pagerank 0.0694
- P_{10} has pagerank 0.1135

□

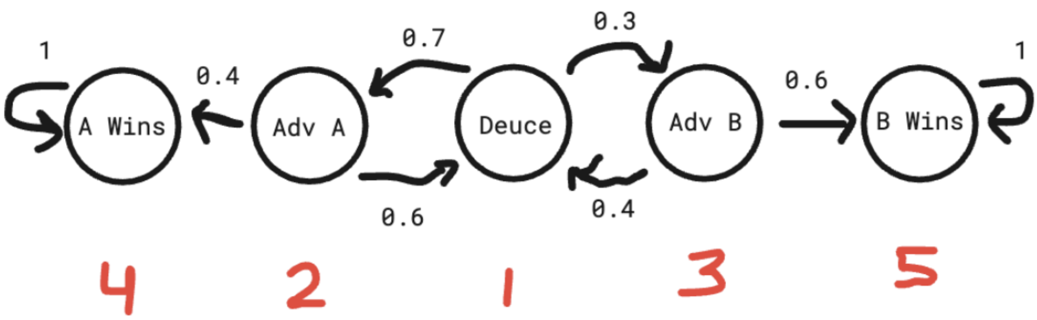
3. Consider the absorbing Markov chain for a tied tennis game from class. It had five states

   1 Deuce (tie)
   2 Advantage A (A has 1 more point than B)
   3 Advantage B (B has 1 more point than A)
   4 A wins.
   5 B wins.

   In this scenario, A is generally a better player than B, but B is more of a "clutch" player, which means that B performs better when the game is on the line. Assume that when the game is tied in Deuce, A has a .7 probability of scoring. But when the game is in either Advantage A or Advantage B, then B has a .6 probability of scoring.

   (a) Draw the diagram for this Markov chain.
   (b) Write down the transition matrix T.
   (c) Assume the game begins in Deuce. Determine which player is more likely to win, and give the probabilities that each player wins.
   (d) Determine the expected number of scores (time steps) that will occur before the game ends (assuming it starts in Deuce).

Solution.
(a)



where the red number indicates the state number in the transition matrix.

$$(b) T = \begin{bmatrix} 0 & 0.6 & 0.4 & 0 & 0 \\ 0.7 & 0 & 0 & 0 & 0 \\ 0.3 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 1 & 0 \\ 0 & 0 & 0.6 & 0 & 1 \end{bmatrix}$$

(c)

$$\lim_{k\to\infty} T^k \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.6087 \\ 0.3913 \end{bmatrix} \implies$$

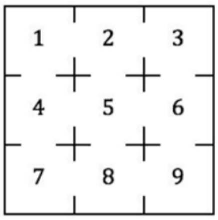
Player A is more likely to win with probability 0.6087.

(d)

$$Q = \begin{bmatrix} 0 & 0.6 & 0.4 \\ 0.7 & 0 & 0 \\ 0.3 & 0 & 0 \end{bmatrix} \implies sum(inv(I-Q)(:,1)) = 4.3478 \text{ scores}$$

□

4. Suppose a mouse moves randomly through the maze below.



As before, we assume that the mouse chooses a new room at each time step, and the mouse is equally likely to any of the available exits. Suppose there is a cat in room 3 and a piece of cheese in room 9. Neither the cat nor the cheese moves[1]. If the mouse reaches the cheese, then the mouse "wins" and it will remain in room 9 eating its celebratory cheese. If the mouse enters the cat's room, then the mouse "loses", as it will be promptly eaten. Consequently, it will remain in room 3 forever. We'll assume that the mouse begins in room 1.

(a) Identify which rooms correspond to transient states and which rooms correspond to absorbing states.

(b) Set up the transition matrix T for this absorbing Markov chain. Be sure to order your states so that all transient states are before the absorbing states. This is necessary so that T has the form $T = \begin{bmatrix} Q & 0 \\ R & I \end{bmatrix}$.

(c) Find the probability that the mouse reaches the cheese.

(d) Determine the expected number of moves that the mouse will make before this game of cat and mouse ends.

Solution.

(a) Rooms 3 and 9 are absorbing states, and the rest are transient states. To ensure that all of the transient states are before the absorbing states, we will swap rows 3 and 8 after finding our initial transition matrix T.

$$(b)\begin{bmatrix}0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0\\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0\\ 0 & \frac{1}{3} & 1 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0\\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{2} & 0 & 0\\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0\\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0\\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{2} & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 1\end{bmatrix} \implies T = \begin{bmatrix}0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0\\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0\\ 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0\\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{2} & 0 & 0\\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0\\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{2} & 1 & 0\\ 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 1\end{bmatrix}$$

Note that now T is in the proper form.

$$\lim_{k\to\infty} T^k x_0 = \begin{bmatrix}0 & \cdots & 0.7059 & 0.2941\end{bmatrix}^T$$

where $x_0 = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T$ because we start in room 1.
(c) Therefore, the probability that the mouse wins and gets the cheese is 0.2941.
(d) Finally, we can find the expected number of moves by the sum of the first column of $(I-Q)^{-1}$ which is 8.0168.

$\square$

5. Now we're going to model (a very simplified version of) the popular childrens' board game Snakes & Ladders as a Markov chain. For this model, we'll use a 3×3 board, with squares numbered 1-9. There will only be one player, who begins the game on square 1. The goal of the game is to land on square 9. The game progresses as follows: on each turn, the player rolls a die and randomly gets a number 1, 2, or 3. The player advances that many spaces. Here are the exceptions: if you land on the base of a ladder, you must climb the ladder to advance to the square on which it ends. If you land on the top of a snake, you must slide down the snake to its bottom. You must land on the exact square to climb/slide the ladder/snake. We will assume there is one ladder going from square 4 to square 7 and there is one snake from square 6 to square 3. To win, you must land exactly on square 9. So if you are on square 7 and roll a 3, you remain on square 7. If you are on square 8 and roll a 2 or 3, you remain on square 8.

   (a) First create a Markov chain for this game without the snake or the ladder. Be clear about what your transition matrix is. Determine the expected number of turns before the game ends.

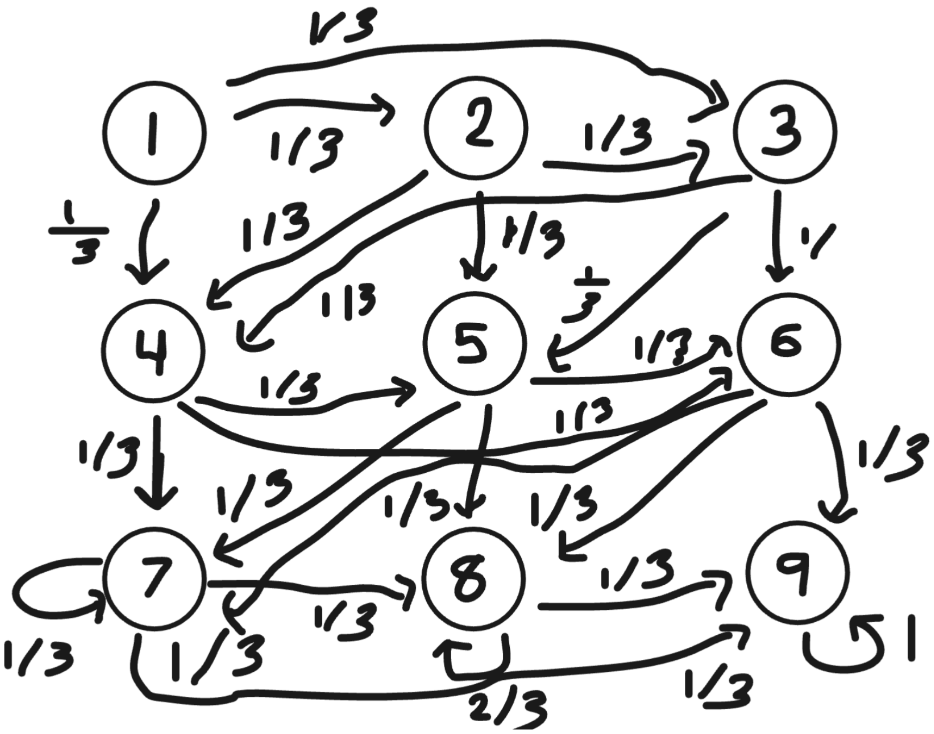
   (b) Now create a Markov chain for this game with the snake and ladder. Determine the expected number of turns before the game ends. (Hint: "being on square 4" and "being on square 6" shouldn't be states in your Markov chain, since it is impossible to remain on those squares.)

   (c) Move the snake so that it goes from square 8 to square 5 (instead of from square 6 to square 3). Note it is the same penalty of -3 spaces as the previous snake. First make a prediction as to whether this increase or decrease the expected number of turns before the game ends. Then compute the expected number of turns.

   Remark: The real Snakes & Ladders has a 10×10 board with 100 spaces and several snakes/ladders. To move, the player rolls a 6-sided die.

Solution.

a. Markov chain without snakes and ladders

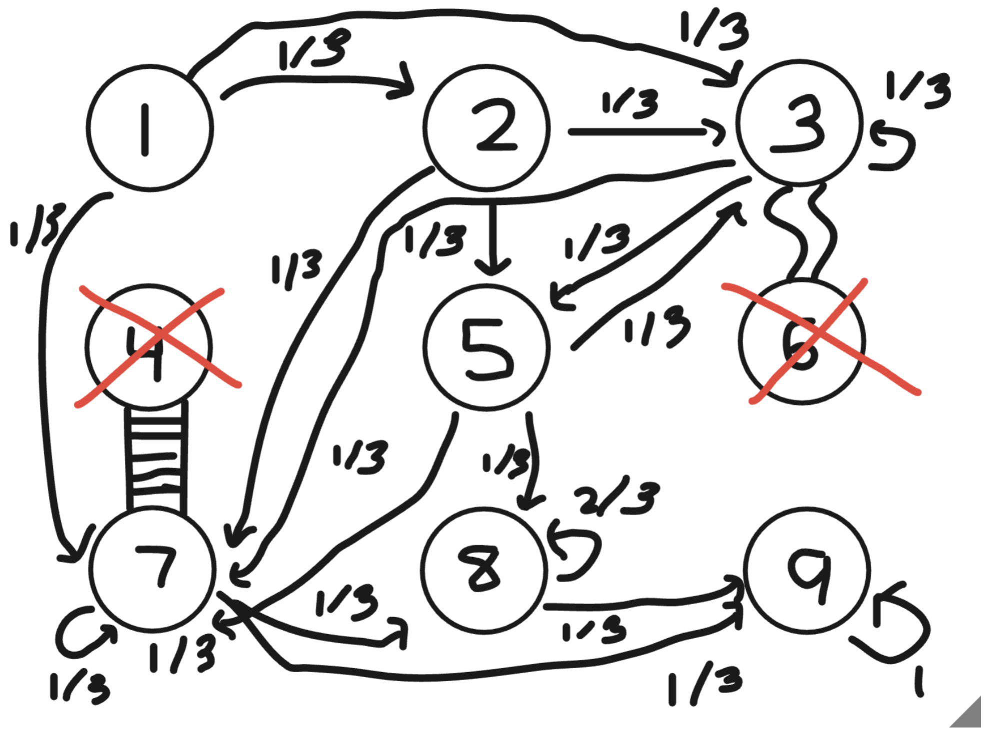


```
T =

         0         0         0         0         0         0         0         0         0
    0.3333         0         0         0         0         0         0         0         0
    0.3333    0.3333         0         0         0         0         0         0         0
    0.3333    0.3333    0.3333         0         0         0         0         0         0
         0    0.3333    0.3333    0.3333         0         0         0         0         0
         0         0    0.3333    0.3333    0.3333         0         0         0         0
         0         0         0    0.3333    0.3333    0.3333    0.3333         0         0
         0         0         0         0    0.3333    0.3333    0.3333    0.6667         0
         0         0         0         0         0    0.3333    0.3333    0.3333    1.0000
```

Thus, by taking the sum of the first column of $(I-Q)^{-1}$, we obtain he expected number of games before the game is over is 5.8272.

b. Markov chain with snakes and ladders




```
T =

         0         0         0         0         0         0         0
    0.3333         0         0         0         0         0         0
    0.3333    0.3333    0.3333    0.3333         0         0         0
         0    0.3333    0.3333         0         0         0         0
    0.3333    0.3333    0.3333    0.3333    0.3333         0         0
         0         0         0    0.3333    0.3333    0.6667         0
         0         0         0         0    0.3333    0.3333    1.0000
```

The expected number of moves in this game is 5.6.

c. I predict that this will increase the expected number of turns because now you are more likely to have to step back when you are closer to the 9. The new expected number of turns is 6.6667.

□