# Homework 3

## Question 1 (Medium)

Figure 1 is a diagram of a maze used in a laboratory experiment. The experiment is begun by placing a mouse at one of the ten interior intersections of the maze. Once the mouse emerges in the outer corridor, it cannot return to the maze. When the mouse is at an interior intersection, its choice of paths is assumed to be random. What is the probability that the mouse will emerge in the "food corridor" when it begins at the $i$th intersection?

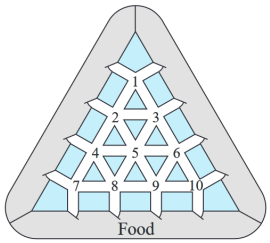


Figure 1

### Hint:

Let the mouse be at intersection 1, the probability of winning starting at this intersection is $p_1$. Starting from intersection 1, the probability of choosing the upper right path is $1/4$ but it will lose because it is not the food corridor, so the probability of winning is $(1/4) * 0$, and the probability of winning by choosing the upper left path is also $(1/4) * 0$ and the probability of winning by choosing the lower right path is $(1/4) * p_3$ (that we do not know yet) and similarly, the probability of winning by choosing the left lower path is $(1/4) * p_2$. Now, we can write the probability of winning starting at intersection 1 $p_1$ as following:

$$p_1 = (1/4)(0) + (1/4)(0) + (1/4)p_2 + (1/4) * p_3$$

Now make the variables $p_i$ on one side and all scalars on the other side and you get:

$$4p_1 - p_2 - p_3 = 0$$

You repeat this step for all 10 intersections and build a system of linear equations. Use Jacobi's method and Gauss-Seidel method to find the vector $\mathbf{p}$ which represents the ten probabilities of winning starting from each intersection. Which method converges faster?

## Question 2 (Easy)

Use the power method with scaling to find a stable eigenvector and eigenvalue of the following matrices:

**(1)** $A = \begin{bmatrix} 0 & 1 & 2 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \end{bmatrix}$ **(2)** $A = \begin{bmatrix} 0 & 6 & 8 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{bmatrix}$

Check if the calculated eigenvalue and eigenvector are correct.

## PageRank Additional Materials

Scikit-Network is a Python library for graph computations. It consists of many algorithms related to graphs. You can find the documentation at the following link:

https://scikit-network.readthedocs.io/en/latest/index.html

One of these algorithms is the PageRank algorithm. You can see an example of using this algorithm to compute the rankings of different pages using the Power Method in the following notebook:

https://github.com/sknetwork-team/scikit-network/blob/master/docs/tutorials/ranking/pagerank.ipynb