Question 1:

Consider a standard chessboard with an 8 × 8 grid of possible locations. We define a Markov chain by randomly moving a single chess piece on this board. The initial location $X_0$ is sampled uniformly among the $8^2 = 64$ squares. At time $t$, the piece then chooses $X_{t+1}$ by sampling uniformly from the set of legal moves given its current location $X_t$. For a description of legal chess moves, see: http://en.wikipedia.org/wiki/Rules_of_chess#Basic_moves.

a) Suppose the chess piece is a king, which can move to any of the 8 adjacent squares. Is the Markov chain irreducible? Is the Markov chain aperiodic?

b) Suppose the chess piece is a bishop. Is the Markov chain irreducible? Is the Markov chain aperiodic?

c) Suppose the chess piece is a knight. Is the Markov chain irreducible? Is the Markov chain aperiodic?

Question 2:

A not-very-smart robot is trapped in a maze shaped like a triangle, with vertices labeled $A, B, C$. Starting from vertex $A$, at each time step, the robot moves to one of the other two vertices. The destination vertex is chosen uniformly at random.

a) Let $r_{AA}(n)$ equal the probability that if the robot starts at vertex $A$, and takes $n \geq 1$ steps, it returns to vertex $A$. Compute $r_{AA}(n)$ for $n = 1, 2, 3, 4$. Prove that for all $n \geq 1$, these probabilities can be written as $r_{AA}(n) = x + y \cdot z^n$ for appropriate constants $x, y, z$.

Suppose that the robot is trapped in a maze shaped like a square, with vertices labeled $A, B, C, D$ in clockwise order. Starting from vertex $A$, at each time step, the robot moves to one of the two adjacent vertices. It has equal probability of choosing these two adjacent vertices, and probability zero of jumping directly to the opposite corner of the square.

b) Repeat your analysis from part (a) for the robot’s random walk in the square maze. Discuss similarities and differences from the triangular maze.

c) Suppose that at vertex $C$ of the square maze, there is a trap that the robot can enter but never leave. From each possible starting state, determine the expected number of time steps for the robot to reach this trapping state.
Question 3:

When you type keywords into a search engine, it displays pages containing related terms, but how should the output be ordered? If you search for “Brown University”, you probably expect “www.brown.edu” to be returned as the most relevant webpage. But what if I created a new webpage that simply listed the words “Brown University” over and over, thousands of times? Would it make sense for this alternative page, which lists the Brown University name far more than Brown’s own page, to be the top ranked result?

To address the weaknesses of rankings based solely on word counts, we explore the famous pagerank algorithm, which formed the basis for at least early versions of Google’s search engine. Think of the whole internet as a directed graph where each node is a website, and there is a directed edge between node $i$ and $j$ if and only if website $i$ hyperlinks to website $j$. Intuitively, the pagerank algorithm seeks a ranking for which: i) If a website is linked to by many other websites, then it’s an important website; ii) If a website has only a few links, but those links comes from authoritative sites (such as “www.brown.edu”), then it’s also important; iii) If a website links to a very large number of other websites, then the “importance” it transfers to each individual site is small. The pagerank algorithm uses Markov chains to allow the information provided by a link to implicitly flow both directions.

To illustrate pagerank, imagine a “random surfer” on the internet that starts at some webpage, and sequentially visits other webpages by following hyperlinks. As illustrated in Figure 1, the surfer chooses between the outgoing links from each page with equal probability. We can then define the “importance” of webpage $i$ as the long-term frequency with which this random surfer visits webpage $i$. If a node has $k$ outgoing edges, then the fraction of time a visit to this node is followed by each linked neighbor is only $\frac{1}{k}$. Denoting the state transition matrix by $T$, if the initial location of the surfer is uniform over the $m$ nodes so $\pi_0 = [\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m}]$, the probability of viewing each webpage after $n$ time steps is then $\pi_0 T^n$. If $n$ is large and there are paths between all pairs of nodes, the state probabilities will converge to a stationary distribution $\pi = \pi T$. Sorting these probabilities gives the pagerank.

a) Create the state transition matrix $T$ for the small network of figure 1. What is the equilibrium distribution of this Markov chain? Which webpage has the highest pagerank?

Of course, the structure of real-world networks is more complex than Figure 1. As illustrated in Figure 2, a naive random surfer could get stuck in a “dead end” page (an absorbing state).
or some locally connected subset of the full web. Thus at each step, with probability $\alpha$ the surfer “teleports” to an arbitrary website at random; each of the $m$ websites has probability $\frac{1}{m}$ of being chosen. With probability $1 - \alpha$, the surfer follows a hyperlink as above. The overall state transition matrix $G$ of this new Markov chain is then

$$G = (1 - \alpha)T + \alpha B,$$

$$B = \frac{1}{m} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}. \quad (1)$$

The matrix $B$ is also known as the “Google matrix”. In our experiments, we set $\alpha = 0.15$.

We now explore a larger network of $m = 9662$ websites. We provide the link structure of those sites in the matrix $L$, where $L(i,j) = 1$ if there is a link from website $i$ to website $j$, and $L(i,j) = 0$ otherwise. The `name` variable stores the names of each website.

b) Write code that creates the state transition matrix $T$, and Google matrix $G$, for the provided data. If website $i$ has no outgoing links, then $T(i,i) = 1$. You should double-check that for both $T$ and $G$, the sum of the transition probabilities for each state equals 1.

c) From a uniform initial state distribution $\pi_0$, apply the Google matrix $G$ for $n = 100$ time steps to compute $\pi_1, \pi_2, \ldots, \pi_{100}$. At each iteration $t$ also compute the magnitude of the absolute change in state probabilities:

$$\epsilon(t) = \sum_{i=1}^{m} |\pi_{t,i} - \pi_{t-1,i}|$$

Plot $\epsilon(t)$ versus $t$. Does $\pi_t$ appear to converge to a limit? If so, which 25 webpages have the highest pagerank? What is the steady-state probability of visiting these top 25 sites? Discuss your results, keeping in mind that this dataset was collected in 2002.

d) Consider a modified pagerank algorithm where instead of following “outgoing links”, the random surfer follows “incoming links”. If the surfer is at website $i$, it chooses uniformly among the sites that link to website $i$. This algorithm is equivalent to reversing the direction of all links in the original data, and then applying the standard pagerank algorithm. Report the top 25 webpages, and their steady-state probabilities, for this reversed pagerank algorithm. Discuss differences from part (c). Which ranking seems more sensible?
Question 4:

You’ve been asked to test the performance of a batch of newly fabricated processors. Assuming the processors were correctly manufactured (the null hypothesis), the average time $X$ to complete your test suite is exponentially distributed with mean 1 (exponential parameter $\lambda_0 = 1$). If the equipment at the factory malfunctions (the alternative hypothesis), the average time $X$ is exponentially distributed with mean 50 (exponential parameter $\lambda_1 = 0.02$). You must decide whether or not this batch of processors was correctly manufactured.

a) Suppose that the factory manufactures correctly functioning processors with probability 0.9. As a function of the observed test suite time $X = x$, what decision rule minimizes the probability that you pick the wrong hypothesis?

b) Suppose that you would like to guarantee that in cases where the factory is operating correctly (the null hypothesis is true), the probability of predicting that the processors are defective is at most 0.05. As a function of the observed test suite time $X = x$, what decision rule satisfying this constraint minimizes the probability that you decide defective processors (the alternative hypothesis) are correct?