Math 20
Lab 4
Due: August 24, 2015

Some code reminders:

We can create a list using: `ourlist <- c()`

We can append a new value to the list, say 5, by: `ourlist <- append(ourlist,5)`

We can repeat something 1000 times by putting that code in a for loop:

```
for(i in 1:1000){
#code
}
```

We can use an if statement to control what our code does:

```
if(state==2){
#code
}
```

We can choose a random number in a range using:

```
newstate <- sample(c(1,3,7),1)
```

Some new commands you might find useful in this lab are the `table()` command which counts occurrences of certain values of a list. For instance, `table(data)` will create a table tallying the number of times specific entries occur. We can plot the tallies using `barplot` . For instance `barplot(table(data))` would make a histogram of the elements of `data`.

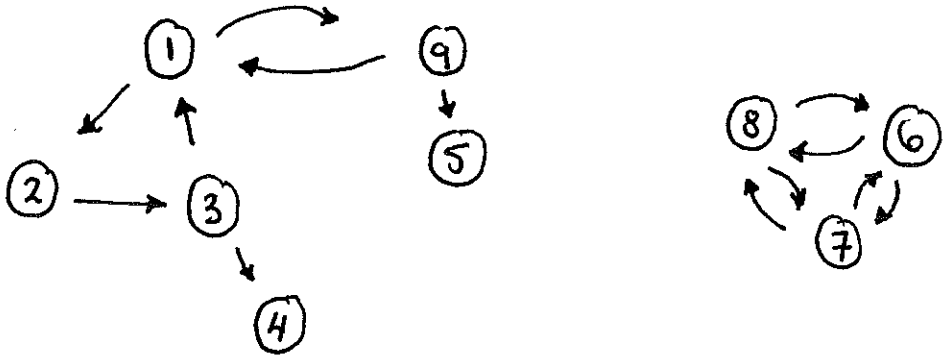
Google's Pagerank algorithm determines how high a ranking a website should get by counting how many different websites link to it. The more links you have, the more important you are, so the higher your rank.

Not all links are counted with equal weight. For instance, our course homepage links to a schedule for the course. That is not as important as a link to the mathematics department's homepage. So in this sense the links to a page are weighted by "importance." Google founders, Brin and Page had the idea of taking all the weighted votes into account using a Markov chain. The chain works by stepping from one website to the next by randomly following one of the links on the page. The amount of time the chain spends on a website tells us how popular a website should be. In this lab we'll explore these ideas by ranking a toy model of the internet which consists of 9 websites.

Note on submission: Following our usual submission protocol, save your files for each task as `task#_yourlastname.R`. To submit the lab, email me the code from each task and answer any questions that are posed in any of the tasks in the body of the email. Note that task 4 does not require any code.

Tasks:

1. Write code that simulates surfing the following model of the internet. Start at a random website, pick one of the links leaving the website at random, and go there. Keep a list of the websites you visit for 50 steps. I'll refer to this action as "surfing the web." Make a plot of your results. Repeat until you have one plot showing the time spent on the left part of the internet and one showing the time spent on the right part.



2. Suppose 1 is your homepage. Simulate surfing the web 1000 times and find out how long it takes on average to end up in an absorbing state. Also keep track of which absorbing state you end up in.

3. Model surfing the web starting at 1 as a Markov chain. Compute the fundamental matrix for this Markov chain. Use this to predict the values from question 2. How do they compare?

4. Obviously there are some problems with this model. We don't like the idea of the absorbing states because once we arrive at one we stay there. So they would have artificially high rank. We can get around this by saying that any time we are in an absorbing state we transition to any website at random on our next step. What would be the long term behavior of this chain? Is there a problem with this?

5. To solve the issue raised in problem 4, we also include a small probability p of transitioning to a random webpage at any state of the chain. Simulate this with $p = 0.1, 0.2, 0.3$ by stepping from site to site 10000 times. Make sure to also get rid of the absorbing states by implementing the fix prescribed in 4. Make a plot of the proportion of time you spend at each website for the different p values. What are the rankings for each of the p values?