Lab 6 - PageRank

From Wikipedia:

PageRank is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. 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.

There are four common frameworks by which academics view Google's PageRank algorithm. The first looks the social impact, both positive and negative, of immediate access to previously unimaginable knowledge through one centralized terminal. The second, and most mathematical, sees PageRank as a computation of the Singular Value Decomposition (SVD) of the adjacency matrix of the graph formed by the internet, with particular emphasis paid the the first few singular vectors. The third, and most far reaching practical, technical implication of Google's work, is the implementation of algorithms and computation at enormous scale. Much of the computing infrastructure which operates at a global scale deployed today can trace its origins to Google's need to perform SVD on an object as enormous as the Internet. Finally, a more intuitive way to look at the PageRank algorithm is through the lens of a web crawler (or many web crawler) acting as an agent (or agents) in a Markov Chain the size of the web. We will investigate this viewpoint.

This crawler is searching for an approximate "invariant" distribution (why does a true invariant distribution almost certainly not exist?) and will rank pages based on their "probability" in this generated distribution. In order to do so, our crawler chooses to follow a link uniformly at random from the page it is on in order to arrive at a new page, keeping tally of how many times it has visited each page. If this crawler runs for a really, really long time, the fraction of time it has spent on each webpage will be approximately the probability of being on that page (assuming we account for pathologies in the Markov chain which we will discuss soon). We then rank pages in decreasing order of probability.
Alright, great! Let’s do stuff. First, visit the following webpage, and see how many web pages can be reached by clicking the links on each page. http://www.eecs.berkeley.edu/~kw1jjang/ee126/1.html

There are total of 8 pages, and they are connected as follows.
Since we choose a link at uniform from each page, the probability of going between pages $x$ and $y$ is 
$\frac{\text{# of pages from } x \text{ to } y}{\text{# of pages leaving } x}$.

Thus the Markov chain generated by the web pages above is

![Markov Chain Diagram]

and the transition matrix of the Markov chain is

$$\begin{pmatrix}
0 & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 & 0 & \frac{1}{5} \\
\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\
\frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\
\frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 & 0 & \frac{1}{5} & 0
\end{pmatrix}$$

1. Find the steady-state (invariant/stationary) distribution $\pi$ of the Markov chain above. How do you know that it exists? The Markov matrix is copied in code below. This might make your computation easier, but you can solve this in any way you wish.

(Note: don't forget about the difference between right and left eigenvectors)

```python
In [13]: import numpy as np
from __future__ import division

P = np.matrix([[0, 1/5, 1/5, 1/5, 0, 0, 1/5],[1/2, 0, 0, 0, 1/2, 0, 0],[0, 1/2, 0, 0, 0, 0, 0],[0, 0, 1, 0, 0, 0, 0],[0, 1/3, 1/3, 1/3, 0, 0, 0],[1/4, 0, 1/4, 0, 0, 1/4, 1/4],[1/4, 1/4, 0, 0, 1/4, 1/4, 0],[1/5, 1/5, 1/5, 1/5, 0, 0, 1/5]])
```
P\_transpose = P.T

print P

\[
[[ 0.  0.2  0.2  0.2  0.2  0.  0. ]
 [ 0.2  ]
 [ 0.5  0.  0.  0.  0.5  0.  0. ]
 [ 0.  ]
 [ 0.  1.  0.  0.  0.  0.  0. ]
 [ 0.  ]
 [ 0.  0.  0.  0.  1.  0.  0. ]
 [ 0.  ]
 [ 0.  0.33333333  0.33333333  0.33333333  0.  0.  0. ]
 [ 0.  ]
 [ 0.25  0.  0.25  0.  0.  0.  0.  0.2 ]
 [ 0.  ]
 [ 0.25  ]
 [ 0.25  0.25  0.  0.  0.25  0.25  0.  0. ]
 [ 0.  ]
 [ 0.2  0.2  0.2  0.2  0.  0.  0.  0.2 ]
 [ 0.  ]]
\]

Simulation time!

We now want to empirically test what we solved above, by modeling a random user hopping along those webpages. We will start the user at "1.html," and behave as per the Markov chain above. In the code below, we simulate this and keep track of the average amount of time a user spends in each state. We will expect that after enough iterations, the fraction of time spent in each state should approach the stationary distribution.

We use the `parse_links()` method to parse all hyperlinks in a page. We use the library Beautiful Soup (http://www.crummy.com/software/BeautifulSoup/) in order to complete this portion of the lab in order to easily parse pages. Once you download the latest release, you must build and install setup.py. Alternatively, use pip or easy_install (help (http://www.crummy.com/software/BeautifulSoup/bs4/doc/)). You are welcome to use different/other tools in your own implementation.
import urlparse
import random
from bs4 import BeautifulSoup

# http://wolfprojects.altervista.org/articles/change-urllib-user-agent/
class MyOpener(urllib.FancyURLopener):
    version = 'Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.15) Gecko/20110303 Firefox/3.6.15'

    # This function will parse a url to give you the domain. Test it!
def domain(url):
        # urlparse breaks down the url passed it, and you split the hostname up
        # Ex: hostname="www.google.com" becomes ['www', 'google', 'com']
        hostname = urlparse.urlparse(url).hostname.split(".")
        hostname = ".".join(len(hostname[-2]) < 4 and hostname[-3:] or hostname[-2:])
        return hostname

    # This function will return all the urls on a page, and return the start url if there is an error or no urls
def parse_links(url, url_start):
        url_list = []
        myopener = MyOpener()
        try:
            # open, read, and parse the text using beautiful soup
            page = myopener.open(url)
text = page.read()
            page.close()
soup = BeautifulSoup(text)

            # find all hyperlinks using beautiful soup
            for tag in soup.findAll('a', href=True):
                # concatenate the base url with the path from the hyperlink
                tmp = urlparse.urljoin(url, tag['href'])
                # we want to stay in the berkeley domain. This becomes more relevant later
                if domain(tmp).endswith('berkeley.edu'):
                    url_list.append(tmp)
                if len(url_list) == 0:
                    return [url_start]
        return url_list
except:
    return [url_start]
In the following code block, we use the above functions to surf the web pages described by the Markov chain above. This code block may take a while to run. If it is taking more than a couple of minutes, maybe try reducing `num_of_visits`.

```python
In [10]:
    import random

    # the url we want to begin with
    url_start = "http://www.eecs.berkeley.edu/~kw1jjang/ee126/1.html"
    current_url = url_start

    # parameter to set the number of transitions you make/different pages you visit
    num_of_visits = 1000

    # dictionary of pages visited so far
    visit_history = {}

    # initialize dictionary since we know exactly where we'll end up
    for i in range(1, 9):
        page = "http://www.eecs.berkeley.edu/~kw1jjang/ee126/" + str(i) + ".html"
        visit_history[page] = 0

    for i in range(num_of_visits):
        # print 'Visiting... ', current_url

        # incrementing the counts
        visit_history[current_url] += 1

        # parsing all the links on the page
        url_list = parse_links(current_url, url_start)

        # returning a random link to go to
        current_url = random.choice(url_list)
```

Does this approximately match the invariant distribution you expected?
Generalizing to the Web

The toy websites given above conveniently form an irreducible Markov chain, but most of the web will not look like this. There will be fringes of the internet containing only self-loops, or some web pages which do not link to others at all. In order to account for such pathologies in the web, we need to make a more intelligent surfer. The simplest idea would be just to jump back to the starting page if there are no links found on the page you are on, and to always return to "good" starting spot with probability \(p\) on every page.

This is a very naive scheme, and there are many more intelligent methods by which you can sample from the distribution of the web, accounting for its pathologies and all. You might want to investigate some of these when working on your own mini-projects.

Ranking Berkeley Professors

The following code is an attempt to rank the Berkeley faculty based on a crawler which begins on the EECS research homepage.

```python
In [4]:
url_start = "http://www.eecs.berkeley.edu/Research/Areas/"
current_url = url_start
num_of_visits = 1000

#List of professors obtained from the EECS page

# bad urls help take care of some pathologies that ruin our surfing
# you might have to be smart with try-catches depending on your application

#Creating a dictionary to keep track of how often we come across a professor
profdict = {}
for i in profs:
    profdict[i] = 0
for i in range(num_of_visits):
```
if random.random() < 0.9: # follow a link!
    print i, ' Visiting... ', current_url
url_list = parse_links(current_url, url_start)
current_url = random.choice(url_list)
    if current_url in bad_urls or "iris" in current_url or "Deptonly" in current_url:
        # dealing with more pathologies
        current_url = url_start
myopener = MyOpener()
page = myopener.open(current_url)
text = page.read()
page.close()
    # Figuring out which professor is mentioned on a page.
    for p in profs:
        profdict[p] += 1 if " " + p + " " in text else 0 # can use regex re.findall(i, text), but it's overkill
else: # click 'home' button!
current_url = url_start

0  Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
1  Visiting...  http://www.eecs.berkeley.edu/XRG/IAB/
2  Visiting...  http://www.eecs.berkeley.edu/department/staff.shtml
4  Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
5  Visiting...  http://www.eecs.berkeley.edu/Research/Areas/OSNT/
6  Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
9  Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
10 Visiting...  http://www.eecs.berkeley.edu/
12 Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
14 Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
15 Visiting...  http://www.eecs.berkeley.edu/
16 Visiting...  http://www.eecs.berkeley.edu/Colloquium/
17 Visiting...  http://www.eecs.berkeley.edu/Rosters/roster.name.nostudentee.html
18 Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
19 Visiting...  http://www.eecs.berkeley.edu/deptinfo
20 Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
21 Visiting...  http://www.cs.berkeley.edu/
22 Visiting...  http://www.eecs.berkeley.edu/department/emergency/sodaill.shtml
23 Visiting...  http://www.eecs.berkeley.edu/department/emergency/sodaill.shtml#finish
25 Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
26 Visiting...  http://www.eecs.berkeley.edu/Research/Areas/SEC/
27 Visiting...  http://www.eecs.berkeley.edu/Research/Areas/
28 Visiting...  http://www.eecs.berkeley.edu/Resguide/stud.shtml
29 Visiting...  http://www.eecs.berkeley.edu/Facilities
30 Visiting...  http://access-guide.berkeley.edu/evacuation-chairs
Visiting... http://recsports.berkeley.edu/outdoor-adventures/classes-clinics/sailing/singlehanded-sailing-clinic/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/education/
Visiting... http://www.eecs.berkeley.edu/education/courses.shtml
Visiting... http://www.eecs.berkeley.edu/Scheduling/EE/schedule-next.html
Visiting... http://www.eecs.berkeley.edu/Courses/Data/59.html
Visiting... http://coe.berkeley.edu/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/XRG/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/SP/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/ENE/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/SP/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Resguide/acad.shtml
Visiting... http://www.eecs.berkeley.edu/Resguide/acad.shtml#top
Visiting... http://facultyguide.berkeley.edu/teaching/menu_1.html
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/department/history.shtml
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/SP/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Rosters/roster.name.nostudentee.html
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/SP/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/department/staffinfo.shtml
Visiting... http://hrweb.berkeley.edu/diversity/staff-eeo
Visiting... http://hrweb.berkeley.edu/glossary/diversity/all
Visiting... http://hrweb.berkeley.edu/sitemap
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/EE/
Visiting... http://www.eecs.berkeley.edu/Gradadm/
Visiting... http://www.eecs.berkeley.edu/MEng/FinancialAid.htm
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/ARC/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/cal/
Visiting... http://www.eecs.berkeley.edu/XRG/entrepreneur.html
630 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
631 Visiting... http://www.eecs.berkeley.edu/Research/Areas/CIR/
632 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
633 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
634 Visiting... http://www.eecs.berkeley.edu/Resguide/acad.shtml
635 Visiting... https://sis.berkeley.edu/bearfactsfacstaff/secure/bfw_domain_menu
636 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
637 Visiting... http://events.berkeley.edu/index.php/calendar/sn/eecs.html?view=quick&timeframe=month&filter=Secondary%20Event%20Type&filtersel=32
638 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
639 Visiting... http://www.eecs.berkeley.edu/Resguide/acad.shtml
640 Visiting... http://www.eecs.berkeley.edu/Resguide/acad.shtml#top
641 Visiting... http://www.eecs.berkeley.edu/Resguide/acad.shtml#top
642 Visiting... http://facultyguide.berkeley.edu/teaching/menu_1.html
643 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
644 Visiting... http://www.eecs.berkeley.edu/Resguide/acad.shtml
645 Visiting... https://wikihub.berkeley.edu/display/calnet/CalNet+Identity+and+Access+Management
646 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
647 Visiting... http://www.eecs.berkeley.edu/Research/Areas/BIO/
648 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
649 Visiting... http://www.eecs.berkeley.edu/XRG/conferences.shtml
650 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
651 Visiting... http://www.eecs.berkeley.edu/Research/Areas/INC/
652 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
653 Visiting... http://www.eecs.berkeley.edu/Research/Areas/COMNET/
654 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
655 Visiting... http://www.eecs.berkeley.edu/Research/Areas/HCI/
656 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
657 Visiting... http://www.eecs.berkeley.edu/Directions/
658 Visiting... http://www.eecs.berkeley.edu/Directions/
659 Visiting... http://www.eecs.berkeley.edu/XRG/recruitment.shtml
660 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
661 Visiting... http://www.eecs.berkeley.edu/department/EECSbrochure/toc.html
662 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
663 Visiting... http://www.eecs.berkeley.edu/education/courses.shtml
664 Visiting... http://www.eecs.berkeley.edu/news/
665 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
666 Visiting... http://www.eecs.berkeley.edu/Research/Areas/Centers/
667 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
668 Visiting... http://www.berkeley.edu/
669 Visiting... http://www.berkeley.edu/life/
670 Visiting... http://www.berkeley.edu/pubserv/
716 Visiting... http://www.eecs.berkeley.edu/bears/
717 Visiting... http://www.eecs.berkeley.edu/XRG/BEARS/2014/presentations/garg.pptx
718 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
719 Visiting... http://www.eecs.berkeley.edu/Resguide/stud.shtml
720 Visiting... http://hkn.eecs.berkeley.edu/
721 Visiting... http://hkn.eecs.berkeley.edu/
722 Visiting... http://hkn.eecs.berkeley.edu/
724 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
725 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
726 Visiting... http://www.eecs.berkeley.edu/help
727 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
728 Visiting... http://www.eecs.berkeley.edu/help
729 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
730 Visiting... http://www.eecs.berkeley.edu/Rosters/roster.name.nostudentee.html
731 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
732 Visiting... http://www.eecs.berkeley.edu/Research/Projects/
733 Visiting... http://www.eecs.berkeley.edu/Research/Projects/Faculty/tomlinn.html
734 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
735 Visiting... http://www.eecs.berkeley.edu/department/people.shtml
736 Visiting... http://www.eecs.berkeley.edu/Students/directories.shtml
737 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
738 Visiting... http://www.eecs.berkeley.edu/Research/Areas/THY/
739 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
740 Visiting... http://www.eecs.berkeley.edu/bears/
741 Visiting... http://www.eecs.berkeley.edu/
742 Visiting... http://www.eecs.berkeley.edu/Faculty/Homepages/maharbiz.html
743 Visiting... http://www.eecs.berkeley.edu/
744 Visiting... http://www.eecs.berkeley.edu/Faculty/Homepages/wainwright.html
745 Visiting... http://www.berkeley.edu/
746 Visiting... http://calmarketplace.berkeley.edu/
747 Visiting... http://calmarketplace.berkeley.edu/whats-new
748 Visiting... http://calmarketplace.berkeley.edu/visit
749 Visiting... http://bcbp.berkeley.edu/atm-locations
750 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
751 Visiting... http://www.eecs.berkeley.edu/Resguide/admin.shtml
753 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
754 Visiting... http://www.eecs.berkeley.edu/Directions/
755 Visiting... http://www.cs.berkeley.edu/Forms/cs.parking.shtml
756 Visiting... http://www.eecs.berkeley.edu/department/staffinfo.shtml
757 Visiting... http://calpact.berkeley.edu/
758 Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/XRG/conferences.shtml
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/department/
Visiting... http://www.eecs.berkeley.edu/Pubs/
Visiting... http://www.eecs.berkeley.edu/Includes/copyright.shtml
Visiting... http://www.eecs.berkeley.edu/cal/
Visiting... http://www.eecs.berkeley.edu/XRG/conferences.shtml
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Gradadm/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/BIO/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/alumni/lists.shtml
Visiting... http://www.eecs.berkeley.edu/Students/lists.shtml
Visiting... http://www.eecs.berkeley.edu/Includes/copyright.shtml
Visiting... http://www.eecs.berkeley.edu/cal/
Visiting... http://www.erso.berkeley.edu/
Visiting... http://engineering.berkeley.edu/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/XRG/IAB/
Visiting... http://www.eecs.berkeley.edu/department/OutreachPrograms.shtml
Visiting... http://nanolab.berkeley.edu/public/general/outreach/summerinternship.pdf
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/XRG/conferences.shtml
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/SCI/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://coe.berkeley.edu/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/department/staff.shtml
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.berkeley.edu/
Visiting... http://trs.p.berkeley.edu/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/deptinfo/login.html?returnto=http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
Visiting... http://www.eecs.berkeley.edu/Faculty/Lists/
Visiting... http://www.eecs.berkeley.edu/Resguide/stud.shtml
Visiting... http://www.eecs.berkeley.edu/XRG/
Visiting... http://www.eecs.berkeley.edu/Research/Areas/
In [5]:
prof_ranks = [pair[0] for pair in sorted(profdict.items(), key=lambda item: item[1], reverse=True)]
for i in range(len(prof_ranks)):
    print "%d: %s" % (i+1, prof_ranks[i])

1: Joseph
2: O'Brien
3: Sastry
4: Goldberg
5: Russell
6: Lee
7: Wu
8: Patterson
9: Katz
10: Fearing
11: Liu
12: Garcia
13: Malik
14: Javey
15: Boser
16: Song
17: Nguyen
18: Maharbiz
19: Bokor
20: Niknejad
21: Hellerstein
<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stojanovic</td>
</tr>
<tr>
<td>Paxson</td>
</tr>
<tr>
<td>Klein</td>
</tr>
<tr>
<td>Culler</td>
</tr>
<tr>
<td>Rabaey</td>
</tr>
<tr>
<td>Chang-Hasnain</td>
</tr>
<tr>
<td>Parekh</td>
</tr>
<tr>
<td>Tomlin</td>
</tr>
<tr>
<td>Brewer</td>
</tr>
<tr>
<td>Salahuddin</td>
</tr>
<tr>
<td>Walrand</td>
</tr>
<tr>
<td>Yelick</td>
</tr>
<tr>
<td>Harvey</td>
</tr>
<tr>
<td>Zakhor</td>
</tr>
<tr>
<td>Fox</td>
</tr>
<tr>
<td>Shenker</td>
</tr>
<tr>
<td>Sangiovanni-Vincentelli</td>
</tr>
<tr>
<td>Anantharam</td>
</tr>
<tr>
<td>Arias</td>
</tr>
<tr>
<td>Seshia</td>
</tr>
<tr>
<td>Sanders</td>
</tr>
<tr>
<td>Ramchandran</td>
</tr>
<tr>
<td>Roychowdhury</td>
</tr>
<tr>
<td>Agrawala</td>
</tr>
<tr>
<td>Canny</td>
</tr>
<tr>
<td>Asanović</td>
</tr>
<tr>
<td>Arcak</td>
</tr>
<tr>
<td>Lustig</td>
</tr>
<tr>
<td>Nikolic</td>
</tr>
<tr>
<td>Keutzer</td>
</tr>
<tr>
<td>Kubiatowicz</td>
</tr>
<tr>
<td>Yablonovitch</td>
</tr>
<tr>
<td>Sahai</td>
</tr>
<tr>
<td>Alon</td>
</tr>
<tr>
<td>Demmel</td>
</tr>
<tr>
<td>Franklin</td>
</tr>
<tr>
<td>Bajcsy</td>
</tr>
<tr>
<td>Wawrzynek</td>
</tr>
<tr>
<td>Bachrach</td>
</tr>
<tr>
<td>Abbeel</td>
</tr>
<tr>
<td>Hartmann</td>
</tr>
<tr>
<td>Bodik</td>
</tr>
<tr>
<td>Darrell</td>
</tr>
<tr>
<td>Stoica</td>
</tr>
<tr>
<td>Spanos</td>
</tr>
</tbody>
</table>
67: Sen
68: Tygar
69: Pisano
Mini-project!

Hopefully this lab has given you a sense of some of the elements at play in ranking pages on the internet. The final portion of this lab is going to be a very open-ended mini-project. This project can be done in groups of up to four people.

Imagine it is the early 2000's, and almost all search engines on the web are essentially like the yellow pages, just made for the Internet. They are indexed manually, and results are often heavily influenced by relationships between advertisers and the search engine. For a quickly growing object the size of the Internet, such a naive form of indexing the web couldn't possibly be the best way of doing things.

You are a researcher who has developed some of the ideas behind using a web crawler to index the web more intelligently, and you have this dream where you might turn this into a multi-billion dollar company that knows way too much information about every internet user in the world. However, one of the great challenges of indexing the entire web is scale. The Internet is an enormous graph relative to the memory and computing power available in even the supercomputers at the time. You want to convince your research advisor that your ideas have great merit, and that they deserve to be given the funding to test them at a legitimate scale. To that end, you want to put together a convincing demo for your research advisor that the potential of indexing the web in this way is enormous, and that it deserves further investigation. (For some background you might want to read this paper [HTTP://INFOLAB.STANFORD.EDU/~BACKRUB/GOOGLE.HTML] to get a feel for where they were coming from when this all began).

Below are some default ideas for a demo you might want to implement, but you should feel free to be creative and come up with something else.

1. Improve on the algorithm to rank Professors in some way that makes it less naive and more effective
2. Rank popularity of fields of research based on which are receiving more attention in papers, awards, etc.
3. Do something interesting with Google Scholar citations (I know the idea of you being a researcher before Google doesn’t make sense anymore if you do this, but whatever.)

Bonus points if you do something that makes fun of Stanford in some way.

You have two weeks to do this mini-project, so it will be due Thursday, October 25th. This project is not intended to be stressful, and is not intended to bring anybody’s grade down too much. It is more of an opportunity for people who are more hands-on engineering oriented to get hacking using some of the ideas we’ve discussed in class so far and use them to do something neat. There is no requirement for how this project ought to be turned in (i.e. it doesn’t have to be an iPython notebook like this and it doesn’t have to be in Python at all), but be smart about how you message your project. You should turn in your code and a written component explaining your project and your results.

** Remember the constraints of time and memory when deciding what to do. The Internet is big. Do something manageable **

Be creative. Be ambitious. Have fun.
Extra credit: prove that finding the first left singular vector of the Markov matrix (generated as described above by picking links uniformly) is equivalent to finding the first left singular vector of the adjacency matrix of a graph. Argue that the crawling technique above will also arrive at the same distribution, assuming we avoid pathologies in the graph.

E.C. Solution