Topics for Today

- Plotting with VPython
- Start Web Surfer Discussion
  - Simple, but powerful model
  - Example of a simulation
- Review on Wednesday
  - Review material and sample questions posted on Blackboard

Plotting in VPython

VPplot1.py
VPplot2.py

```
from numpy import *
from visual.graph import *          # import graphing features

funct1 = gcurve(color=color.cyan)

for x in arange(0., 8.1, 0.1):      # x goes from 0 to 8
    funct1.plot(pos=(x, 5.*cos(2.*x)*exp(-0.2*x)))
                                    # add point to plot
```

pos=(x,y) adds points to the plot shown in the display

Clicker Question

Which web search engine do you use the most?

A. **Yahoo!**
B. **Google**
C. **Altavista**
D. **ChaCha**
E. **Other**

What makes Google so effective and gives it over 50% of the market share?

- Larry Page and Sergey Brin are smart and creative --- and were at the right place at the right time in 1998.
- Developed PageRank for measuring the relative importance of web pages.
  - A link from page A to page B is considered a vote
  - Who page A is, matters: votes cast by important pages count more

The Random surfer

- PageRank measures the likelihood that a person randomly clicking on links will arrive at a particular page.
- A Page Rank is a value between 0 and 10
- Early work on PageRank used a **random surfer model**
- The PageRank of a page was derived from the probability of visiting that page when clicking on links at random.

The Random Surfer

- The random surfer model proved to be surprisingly accurate
  - 90% of the time the random surfer clicks on a random link on the current page
  - 10% of the time he/she goes to a random page

Of course, real users do not randomly surf the web, but follow links according to interest and intention.

Flaws of the model

- Pages are not chosen with equal probability
- The 90-10 breakdown is just a guess
- Bookmarks and back buttons are not considered
- A simulation works on a small size model

Readable reference:

Introduction to Programming in Java: An Interdisciplinary Approach, Robert Sedgewick and Kevin Wayne, Addison Wesley, 2006.

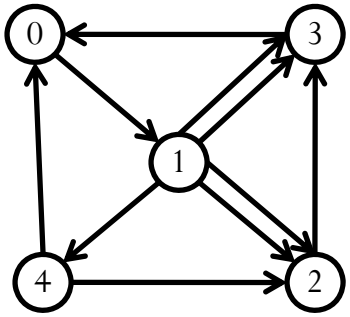
However, ...

Random surfer model allows one to study a number of properties of the web:

- What web page is the random surfer most likely to visit?
- Within a given time, how often does the random surfer visit each page?
- If the sturfer starts at page A, what is probability that he/she ends up at page B after t steps?

Input Representation

- N pages, numbered 0 to N-1
- Represent each link as a pair of integers




```
5
0 1
1 2
1 2
1 3
1 3
1 4
2 3
3 0
4 0
4 2
```

Monday, February 16, 2009

9

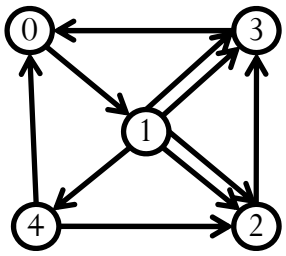
Internal representation

LinkCount Matrix

```
0 1 0 0 0
0 0 2 2 1
0 0 0 1 0
1 0 0 0 0
1 0 1 0 0
```

Degree

```
1
5
1
1
2
```



The degree is the number of links leaving a page.
It is the sum of the entries on a row of the LinkCount matrix

Use list-of-lists or an array consisting of N rows and columns

Monday, February 16, 2009

10

Clicker question

```
from numpy import *

liA = [[1,2,3], [0,0,0], [0,0,0]]
liA[1][2] = 1.2
print liA

arA = zeros((3,3))
arA[0] = range(1,4)
arA[1][2] = 1.2
print  arA
```

What is printed?

A.
[[1, 2, 3], [0, 0, 1.2], [0, 0, 0]]
[[0. 1. 2.], [0. 0. 1.2], [0. 0. 0.]]

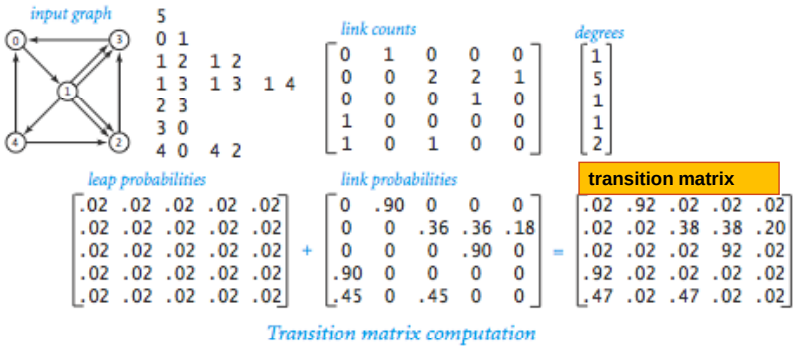
B. - correct answer
[[1, 2, 3], [0, 0, 1.2], [0, 0, 0]]
[[1. 2. 3.]
 [0. 0. 1.2]
 [0. 0. 0.]]

[C.
[1, 2, 3], [0, 0, 1.2], [0, 0, 0]]
[[1 2 3]
 [0 0 1.2]
 [0 0 0]]

Transitions (1)

- If the surfer is at page A, the probability of page B being chosen as a random page is **N/0.10**
  (If N=5, this probability is 0.02)
- If the surfer is at page A and there are degree[A] links to other pages, then the probability that the link to page B is followed is
  LinkCount[A][B] * 0.90 / degree[A]
- The probability of moving from page A to page B, is
  N/0.10 + LinkCount[A][B] * 0.90 / degree[A]

Transitions (2)



Transition matrix computation

The probability of moving from page A to page B, is

N/0.10 + LinkCount[A][B] * 0.90 / degree[A]

Simulation Overview

1. Read the input pairs and generate the transition matrix
2. Start the surfer at current_site = 0
3. Run the simulation for SURF_COUNT moves.
   Each move does the following:
   - Determine a random new_site using the transition matrix
   - Make a move from from current_site to new_site
   - Update any variables that track the simulation

What does the simulation return?

- Count how often each page is visited during the simulation
  - Represented in a list/array of size N
  - Array hit_count
- Show the entries of hit_array as a bar graph
- Wish: Draw a graphical representation of the connection between the pages and visualize where the surfer currently is

Two Functions

def get_transitions(input_file):

takes a file as input

generates the transition_matrix,

returns the transition_matrix and the number_of_pages

def simulate(trans_mat, N, iters):

takes 3 parameters: transition matrix, number of pages, and number of iterations.

Simulates the random surfer making iters moves from page to page; returns a hit_count array

```
def get_transitions(input_file):
    f = open(input_file)
    N = int(f.readline())          # Find out how many pages

    # Create two arrays to store information about the page link structure
    link_counts = zeros((N,N))
    out_degrees = zeros(N)

    # Process link list; every line contains a pair of numbers
    for line in f:
        i,j = line.split()
        i,j = int(i),int(j)
        link_counts[i][j] += 1
        out_degrees[i] += 1
```


def get_transitions(input_file) – continued

```
# Calculate the transition matrix given the 90/10 rule
RANDOM_PROB = .1
transition = zeros((N,N))

for i in range(N):
    for j in range(N):
        transition[i][j] =
            ((1-RANDOM_PROB) * link_counts[i][j] / out_degrees[i]
                    + RANDOM_PROB/N)
return transition, N
```

```
def simulate(trans_mat, N, iters):
    page_hits = zeros(N)
    current_site = 0

    # Surf's up!
    for k in xrange(iters):
        new_site = get_next_page(trans_mat, N, current_site )
        page_hits[new_site] += 1
        current_site = new_site

    return page_hits
```


How do we determine new_site?

Transition Matrix

	0	1	2	3	4
0	0.02	0.92	0.02	0.02	0.02
1	0.02	0.02	0.38	0.38	0.02
2	0.02	0.02	0.02	0.92	0.02
3	0.92	0.02	0.02	0.02	0.02
4	0.47	0.02	0.47	0.02	0.02

Computing new_site

CurrentSite =2

Row 2 of transition matrix: 0.02 0.02 0.02 0.92 0.02

Choose a random number between 0 and 1: 0.85

Cumulated values 0.02 0.04 0.06 **0.98** 1.00

CurrentSite = 4

Row 4 of transition matrix: 0.47 0.02 0.47 0.02 0.02

Random number: 0.48

Cumulated values 0.47 **0.49** 0.96 0.98 1.00


```
def get_next_page(trans_mat, N, current_site):
    r = random.uniform(0,1)
    total = 0
    # start at position 0 in row current_site and add up values
    for k in range(N):
        total = total + trans_mat[current_site][k]
        if total >= r:
            return k
```

Putting it all together …

- File **matrix_surfer.py**
- Use pylab to show the histogram
- For small values of N, show entries computed
- Visualization of the surfer along the links needs different software packages

Questions

- Every time we call get_next_page, we start adding up values until we exceed the generated probability
- This does some re-computations. Can they be avoided?
- Yes.
  If we pre-compute the sums, we should to search for the entry needed in a different way