Announcements

- P6 due today at 6pm
- Final exam:
  - Thurs, 5/8, 9am, Barton East & Central
- Please fill out course evaluation on-line, see "Exercise 15"
- Regular office/consulting hours end tomorrow. Revised hours next week.
- Pick up papers during consulting hours at Carpenter
- Read announcements on course website!

- Previous Lecture:
  - Models and data
    - Congressional apportionment
    - Sensitivity analysis
- Today's Lecture:
  - Simulation—Google "page rank"
  - Optimization—the traveling salesperson problem

Quantifying Importance

How do you rank web pages for importance given that you know the link structure of the Web, i.e., the in-links and out-links for each web page?

A related question:

How does a deleted or added link on a webpage affect its "rank"?

Background

Index all the pages on the Web from 1 to n. (n is around ten billion.)

The PageRank algorithm orders these pages from "most important" to "least important".

It does this by analyzing links, not content.

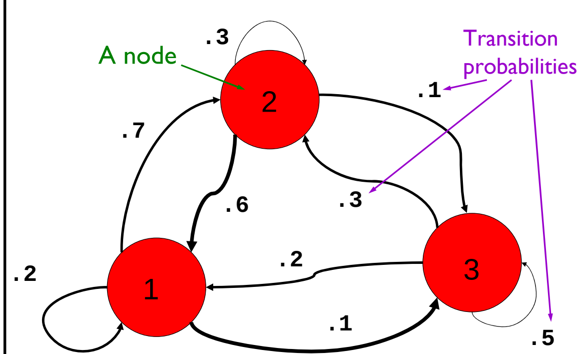
Key Ideas

The Transition Probability Matrix

A very special random walk

The Connectivity Matrix

A 3-node network with specified transition probabilities

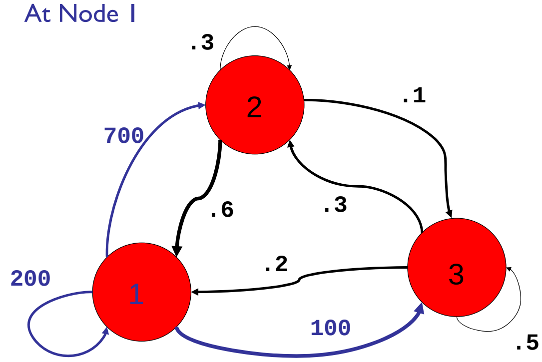


A special random walk

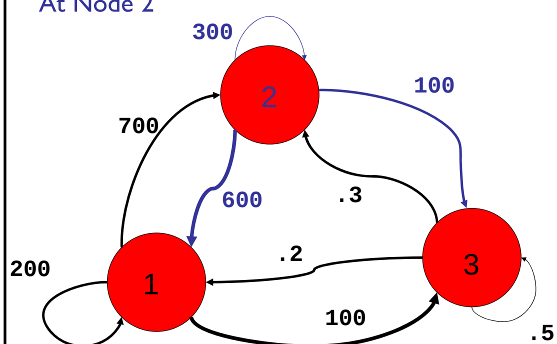
Suppose there are a 1000 people on each node.

At the sound of a whistle they hop to another node in accordance with the "outbound" probabilities.

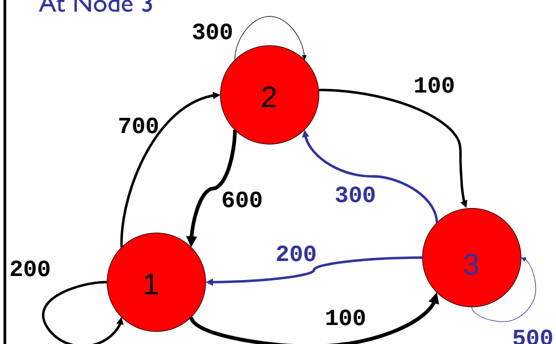
At Node 1



At Node 2



At Node 3



New Distribution

	Before	After
Node 1	1000	1000
Node 2	1000	1300
Node 3	1000	700

Repeat

	Before	After
Node 1	1000	1120
Node 2	1300	1300
Node 3	700	580

State Vector

```
Time 0     [1000 1000 1000]
Time 1 → [1000 1300  700]
Time 2 → [1120 1300  580]
```

The state of each node at a specific time

After 100 Iterations

	Before	After
Node 1	1142.85	1142.85
Node 2	1357.14	1357.14
Node 3	500.00	500.00

Appears to reach a steady state

Call this the stationary vector

Transition Probability Matrix

P

.2	.6	.2
.7	.3	.3
.1	.1	.5

`P(i,j)` is the probability of hopping to node `i` form node `j`

Formula for the new state vector

P

.2	.6	.2
.7	.3	.3
.1	.1	.5

```
w(1) = .2*v(1) + .6*v(2) + .2*v(3)
w(2) = .7*v(1) + .3*v(2) + .3*v(3)
w(3) = .1*v(1) + .1*v(2) + .5*v(3)
```

`v` is the old state vector
`w` is the updated state vector

Formula for the new state vector

P

.2	.6	.2
.7	.3	.3
.1	.1	.5

```
W(1) = P(1,1)*v(1) + P(1,2)*v(2) + P(1,3)*v(3)
W(2) = P(2,1)*v(1) + P(2,2)*v(2) + P(2,3)*v(3)
W(3) = P(3,1)*v(1) + P(3,2)*v(2) + P(3,3)*v(3)
```

`v` is the old state vector
`w` is the updated state vector

The general case

```
function w = Update(P,v)
% Update state vector v based on transition
% probability matrix P to give state vector w
n = length(v);
w = zeros(n,1);
for i=1:n
   for j=1:n
      w(i) = w(i) + P(i,j)*v(j);
   end
end
```

To obtain the stationary vector…

```
function [w,err]= StatVec(P,v,tol,kMax)
% Iterate to get stationary vector w
w = Update(P,v);
err = max(abs(w-v));
k = 1;
while k<kMax && err>tol
      v = w;
      w = Update(P,v);
      err = max(abs(w-v));
      k = k+1;
end
```

May 1, 2008 Lecture 28 30

A random walk on the Web

Repeat:

You are on a webpage.
There are m outlinks.
Choose one at random.
Click on the link.

What if there are no outlinks?
We'll deal with dead ends later.

May 1, 2008 Lecture 28 31

A random walk on the Web

Repeat:

You are on a webpage.
There are m outlinks.
Choose one at random.
Click on the link.

- Need transition probabilities
- Eventually will get to steady state

May 1, 2008 Lecture 28 32

A Connectivity Matrix

`G(i,j)` is 1 if there is a link on page `j` to page `i`. (I.e., you can get to `i` from `j`.)

G

0	1	0	0	1	0	1	0
1	0	0	0	0	0	1	1
0	1	0	0	1	0	0	0
1	0	1	1	0	1	0	0
0	0	0	1	0	0	1	0
0	1	1	0	0	1	0	0
1	0	0	0	0	0	1	0
0	0	1	0	0	1	0	0

May 1, 2008 Lecture 28 33

The Transition Probability Matrix is obtained from the Connectivity Matrix!

a = 1/3

b = 1/2

c = 1/4

G

0	a	0	0	b	0	c	0
a	0	0	0	0	0	c	1
0	a	0	0	b	0	0	0
a	0	a	b	0	a	0	0
0	0	0	b	0	0	c	0
0	a	a	0	0	a	0	0
a	0	0	0	0	0	c	0
0	0	a	0	0	a	0	0

May 1, 2008 Lecture 28 34

Connectivity (G) → Transition Probability (P)

```
[n,n] = size(G);
P = zeros(n,n);
for j=1:n
   P(:,j) = G(:,j)/sum(G(:,j));
end
```

May 1, 2008 Lecture 28 35

Connectivity

0	0	0	0	0	0	1	1
1	0	0	1	0	0	0	0
1	0	1	0	0	1	0	1
0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0

Transition Probability

0	0	0	0	0	0	1	.25
.33	0	0	.50	0	0	0	0
.33	0	.25	0	0	1	0	.25
0	0	0	0	1	0	0	0
.33	0	.25	0	0	0	0	.25
0	0	.25	0	0	0	0	.25
0	0	.25	0	0	0	0	0
0	1	0	.50	0	0	0	0

May 1, 2008 Lecture 28 36

Stationary vector represents how "popular" the pages are → PageRank

statVec	sorted	idx	pR
0.5723	0.8911	6	4
0.8206	0.8206	2	2
0.7876	0.7876	3	3
0.2609	0.5723	1	6
0.2064	0.4100	8	8
0.8911	0.2609	4	1
0.2429	0.2429	7	7
0.4100	0.2064	5	5

May 1, 2008 Lecture 28 37

```
[sorted, idx] = sort(-statVec);
for k= 1:length(statVec)
   j = idx(k) % index of kth largest
   pR(j) = k;
end
```

statVec	sorted	idx	pR
0.5723	0.8911	6	4
0.8206	0.8206	2	2
0.7876	0.7876	3	3
0.2609	0.5723	1	6
0.2064	0.4100	8	8
0.8911	0.2609	4	1
0.2429	0.2429	7	7
0.4100	0.2064	5	5

May 1, 2008 Lecture 28 38

A new random walk on the Web that deals with dead ends

```
Repeat:
    You are on a webpage.
    If there are no outlinks
       Pick a random page and go there.
    else
       Flip an unfair coin.
       if heads
          Click on a random outlink and go there.
       else
          Pick a random page and go there.
       end
    end
```

In practice, an unfair coin with prob .85 heads works well.

This results in a different transitional probability matrix.

May 1, 2008 Lecture 28 39

Connectivity

Page 6 has no outlinks.

0	0	0	0	0	0	1	1
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0

Transition Probability

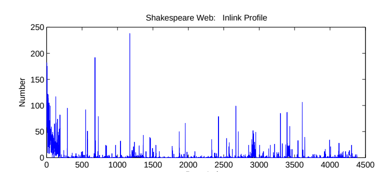
0	0	0	0	0	?	1	.25
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	?	0	0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	?	0	.25
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	?	0	0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	?	0	.25
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	?	0	.25
0	0	.25	0	0	?	0	0
0	1	0	.50	0	?	0	0

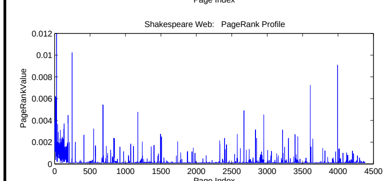
A. 0
B. 0.125
C. 1
D. rand

Other columns need to be changed as well. See **PageRank.m**

May 1, 2008 Lecture 28 40

Shakespeare SubWeb (n=4383)

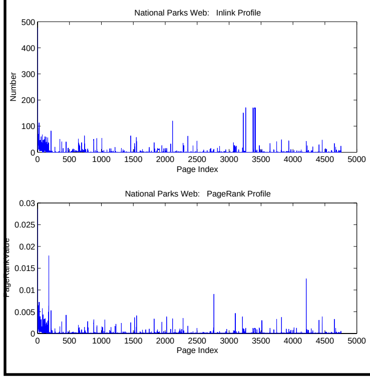




PRank	InRank
1	24
2	417
3	110
4	14
5	68
6	8
7	37
8	54
9	2
10	261
11	1
12	67
13	118
14	50
15	3

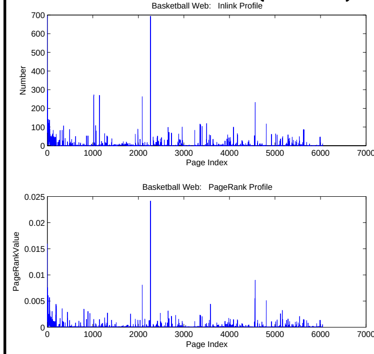
41

Nat'l Parks SubWeb (n=4757)



PRank	InRank
1	1
2	100
3	77
4	386
5	62
6	110
7	37
8	109
9	127
10	32
11	28
12	830
13	169
14	168
15	64

Basketball SubWeb (n=6049)



PRank	InRank
1	2
2	1
3	20
4	19
5	3
6	61
7	23
8	43
9	91
10	28
11	85
12	358
13	313
14	71
15	68

Optimization

- Find the "best" of something
  - the shortest path
  - the most cost efficient production line
  - the lowest-risk investment strategy
- There is a search (solution) space
- There is some kind of objective function
- There are usually constraints
- Usually willing to accept suboptimal solution if it is "good enough" and is cheap to compute

The Traveling Salesperson Problem (TSP)

A salesperson must travel to a set of cities exactly once and then return to the starting city. What is the shortest path?

Just enumerate all possible paths and pick shortest?

(n-1)! possibilites

4 cities

1	2	3	4
1	2	4	3
1	3	2	4
1	3	4	2
1	4	2	3
1	4	3	2

5 cities

1	2	3	4	1	4	2	3
1	2	3	5	1	4	2	5
1	2	4	3	1	4	3	2
1	2	4	5	1	4	3	5
1	2	5	3	1	4	5	2
1	2	5	4	1	4	5	3
1	3	2	4	1	5	2	3
1	3	2	5	1	5	2	4
1	3	4	2	1	5	3	2
1	3	4	5	1	5	3	4
1	3	5	2	1	5	4	2
1	3	5	4	1	5	4	3

Just enumerate all possible paths and pick shortest?

(n-1)! possibilites

If a computer can process 1 billion itineraries a second, how long does it take to solve a 100-city TSP?

About a century…

A heuristic is a computational rule-of-thumb that points us towards optimality but without any guarantee that optimality will actually be achieved.

- A heuristic for the TSP:

  From the current location, choose to visit the *nearest unvisited* city

May 1, 2008 Lecture 28 48

Organization of the TSP program

```
% Visit n cities, starting from city 1
Put cities 2:n in unvisited list
for k= 2:n
   Find nearest unvisited city, c
   Put city c in the tour path
   Remove city c from unvisited list
end
Return to city 1
```

May 1, 2008 Lecture 28 49

What we learned…

- Develop/implement algorithms for problems
- Develop programming skills
  - Design, implement, document, test, and debug
- Programming "tool bag"
  - Control flow (if-else; loops)
  - Functions for reducing redundancy
  - Data structures
  - Graphics
  - File handling

May 1, 2008 Lecture 28 50

What we learned… (cont'd)

- Applications and concepts
  - Image and sound
  - Sorting and searching
  - Divide-and-conquer strategies
  - Approximation and error
  - Simulation and optimization
  - Computational effort and efficiency

May 1, 2008 Lecture 28 51

Final Exam

- Thurs 5/8, 9-11:30am, Barton East and Central.
- Covers entire course, but emphasizes material after Prelim 3
- Closed-book exam, no calculators
- Bring student ID card

- Check for announcements on webpage:
  - Study break office/consulting hours
  - Review session time and location
  - Review questions
  - List of potentially useful functions

May 1, 2008 Lecture 28 53