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

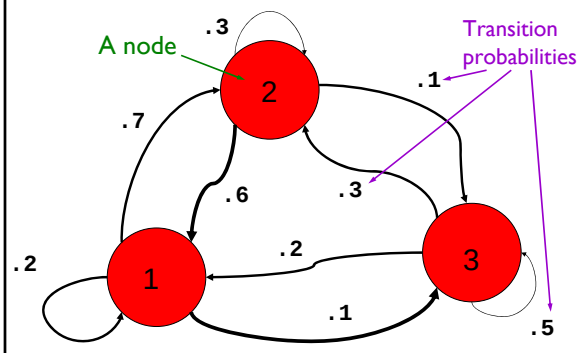
Background

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.

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.

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

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
```


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.

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
```


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
```

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
```


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

```
0.5723
0.8206
0.7876
0.2609
0.2064
0.8911
0.2429
0.4100
```
statVec

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

```
4
2
3
6
8
1
7
5
```
pR


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

```
0.5723
0.8206
0.7876
0.2609
0.2064
0.8911
0.2429
0.4100
```
statVec

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

```
4
2
3
6
8
1
7
5
```
pR

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.

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
 0   0   0   0   0   ?   0   0
 0   0   0   0   0   ?   0  .25
 0   0   0   0   0   ?   0   0
 0   0   0   0   0   ?   0  .25
 0   0   0   0   0   ?   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`

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?

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
```


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

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

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

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