

# Announcements

- **P6 due today** at 11pm
- **Final exam:**
  - Thurs, 5/12, 9am, **Barton East (indoor field)**
- Please fill out course evaluation on-line, see “Exercise 15”
- Regular office/consulting hours end tonight. Revised hours next week.
- Pick up papers during consulting hours at Carpenter
- **Read announcements on course website!**

- Previous Lecture:

- Recursion review
- Efficiency

- 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.)

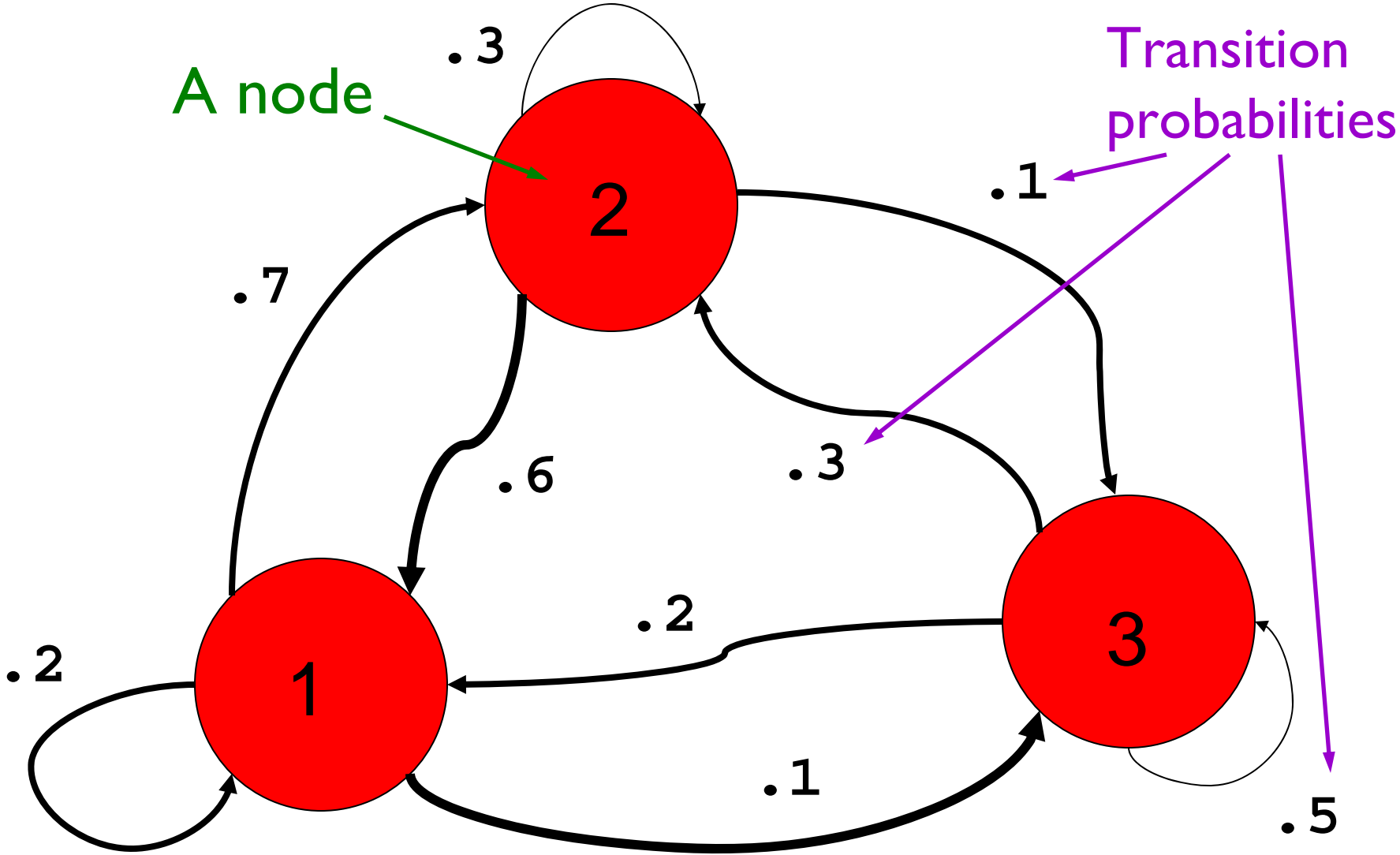
The **PageRank** algorithm orders these pages from “most important” to “least important.”

It does this by **analyzing links, not content.**

## Key ideas

- There is a random web surfer—a special **random walk**
- The surfer has some random “surfing” behavior—a **transition probability matrix**
- The transition probability matrix comes from the link structure of the web—a **connectivity matrix**
- Applying the transition probability matrix → **Page Rank**

# A 3-node network with specified transition probabilities



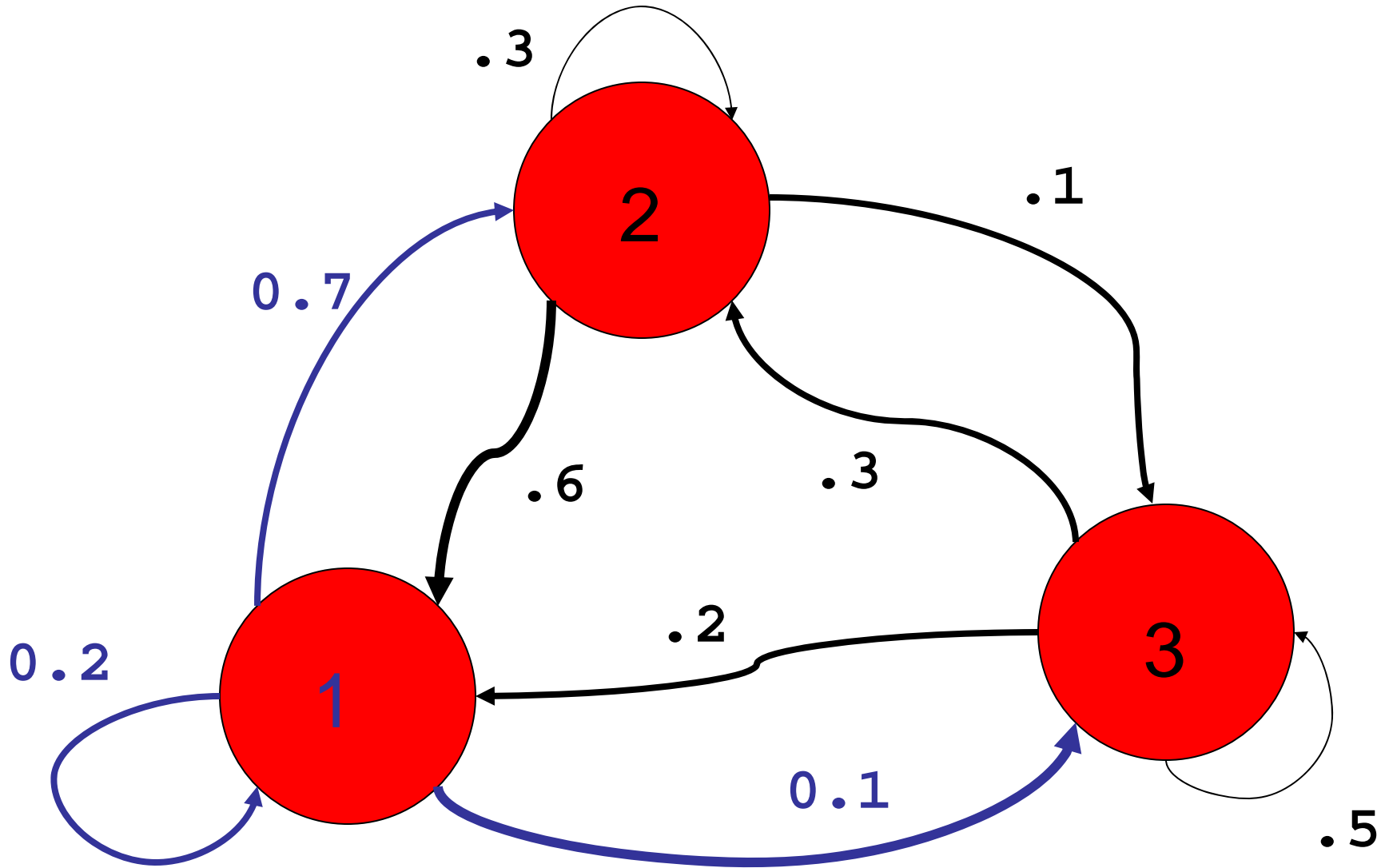
## A special random walk

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.

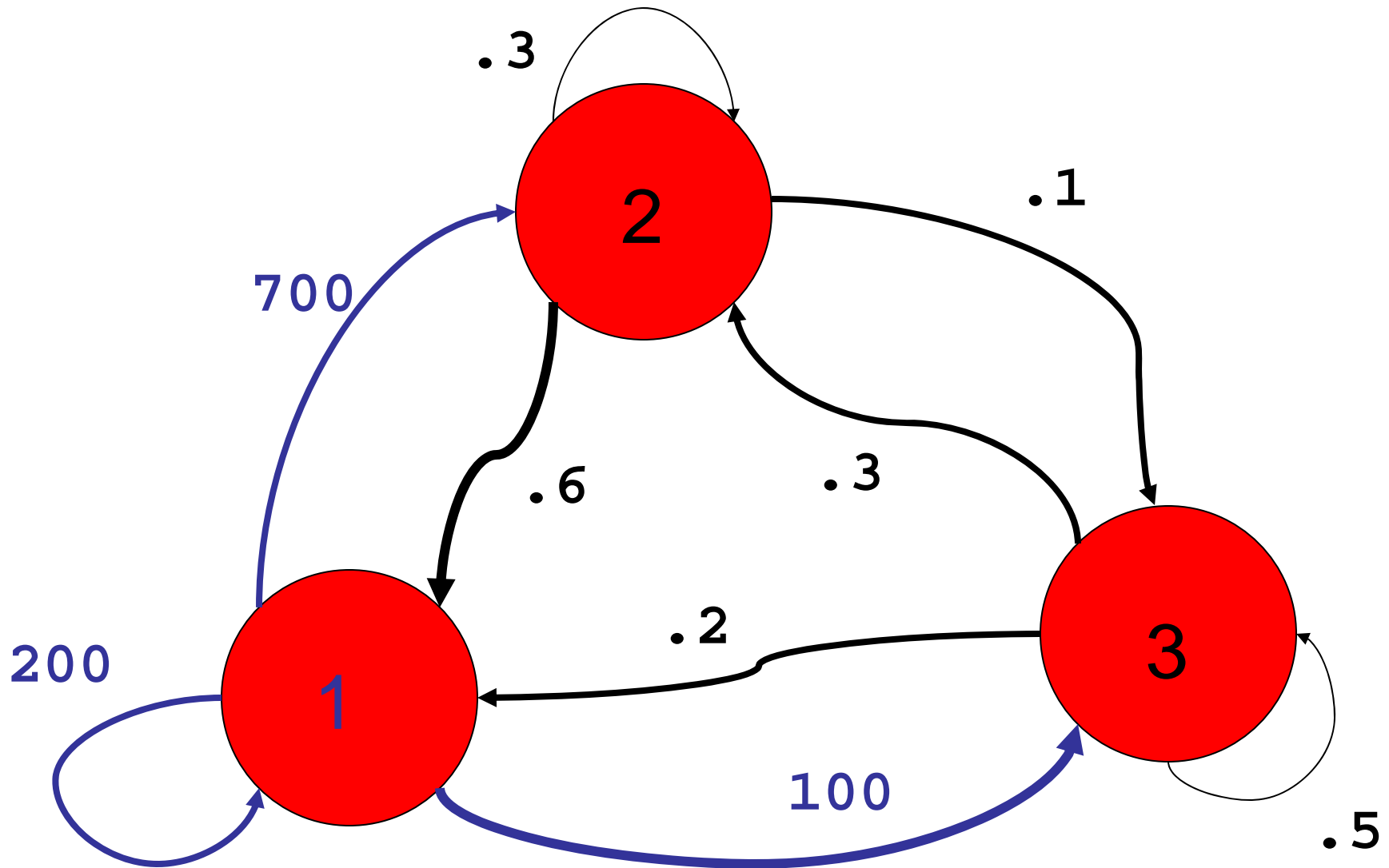
For now we assume we know these probabilities. Later we will see how to get them.

At Node 1

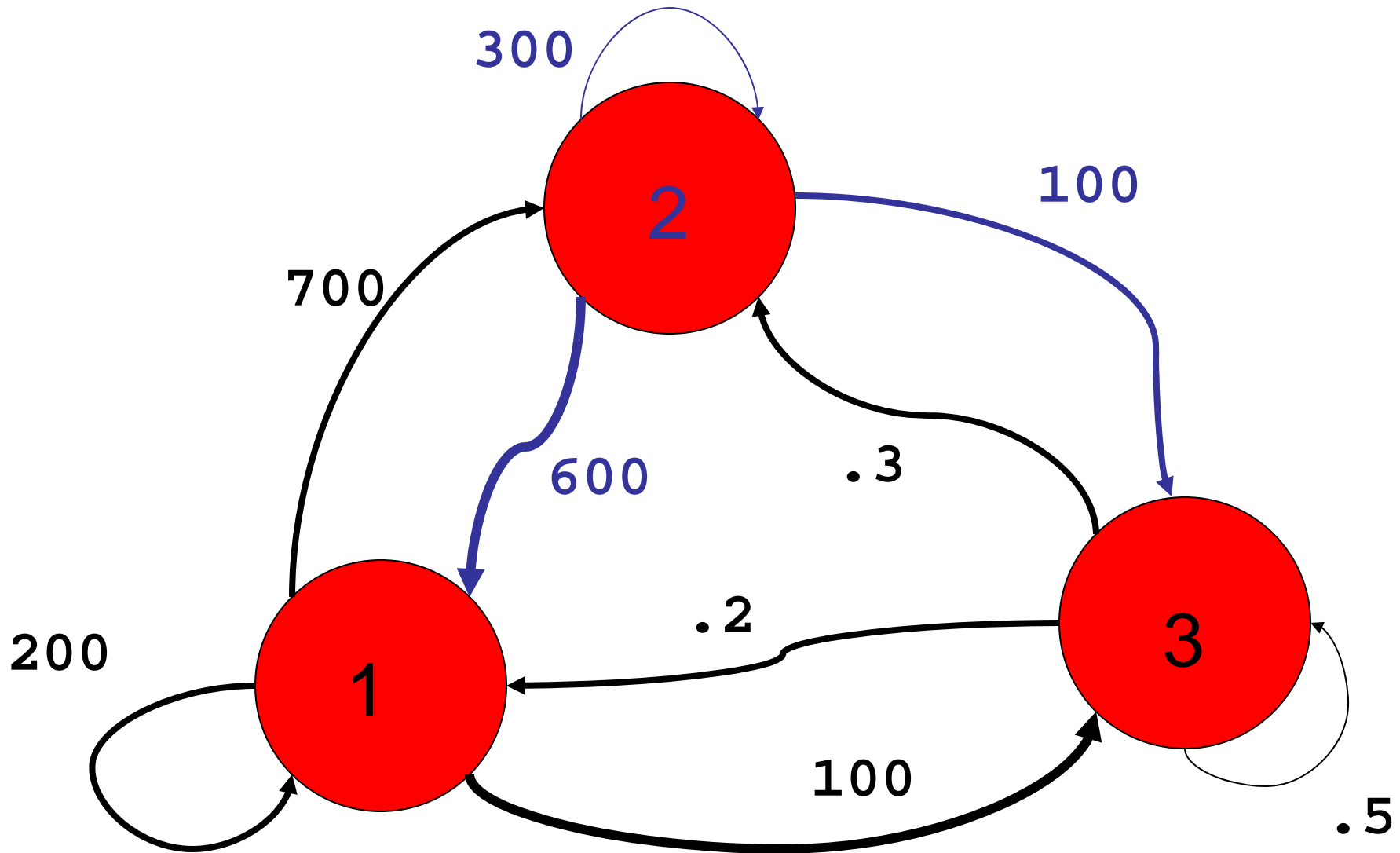




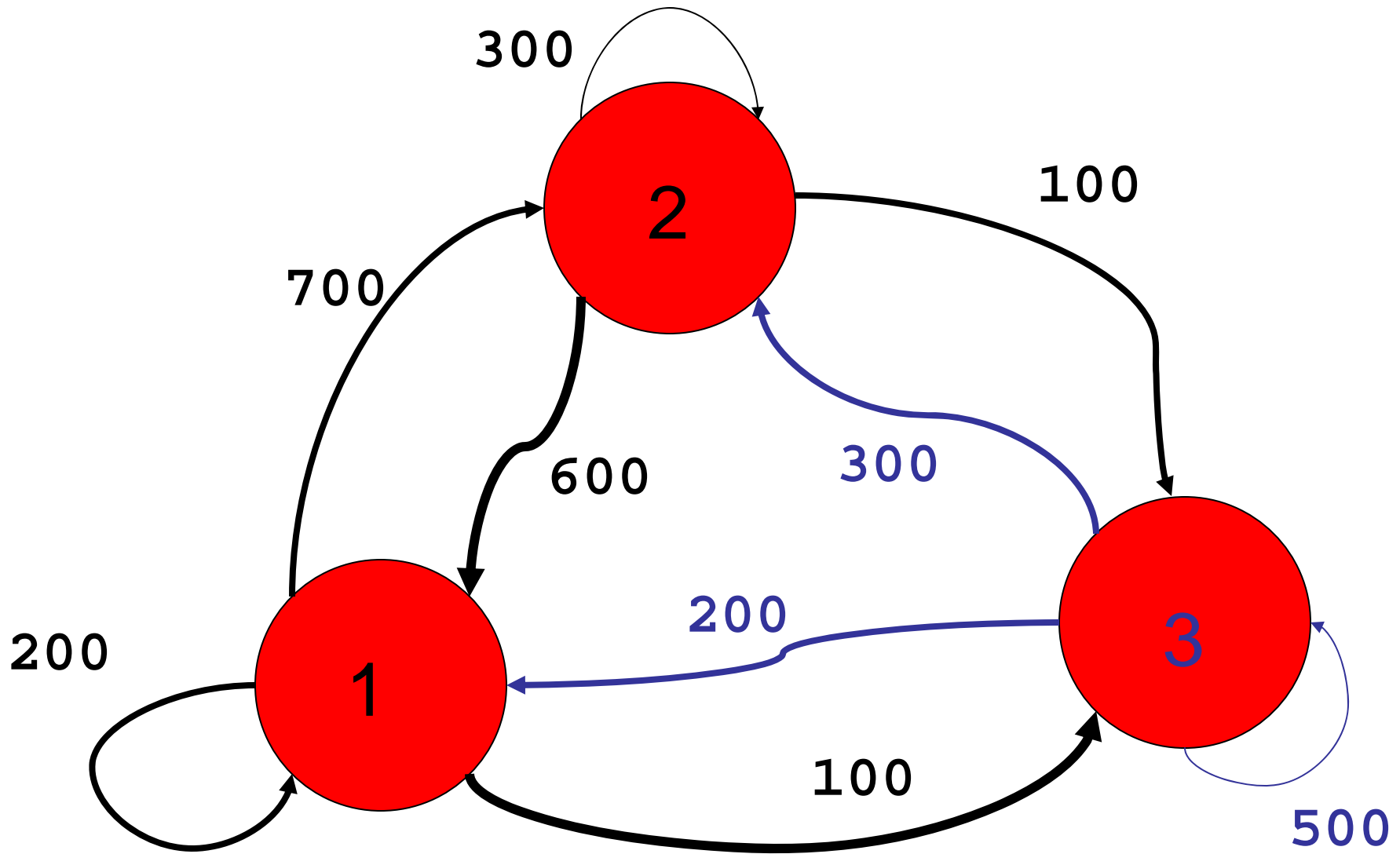
At Node 1



# At Node 2



# At Node 3



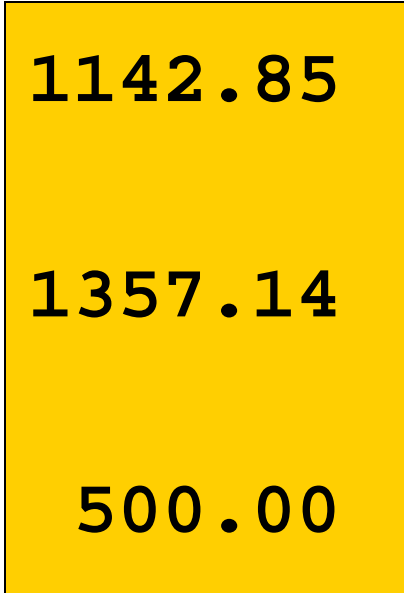
# State Vector:

describes the state at each node at a specific time

$$\begin{array}{ccc} T=0 & T=1 & T=2 \\ \left( \begin{array}{c} 1000 \\ 1000 \\ 1000 \end{array} \right) & \left( \begin{array}{c} 1000 \\ 1300 \\ 700 \end{array} \right) & \left( \begin{array}{c} 1120 \\ 1300 \\ 580 \end{array} \right) \end{array}$$

## After 100 iterations

	<b>T=99</b>	<b>T=100</b>
<b>Node 1</b>	1142.85	1142.85
<b>Node 2</b>	1357.14	1357.14
<b>Node 3</b>	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  $i$  from node  $j$

## Formula for the new state vector

$P$

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

$P(i, j)$  is  
probability of  
hopping to node  
 $i$  from node  $j$

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

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

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

$v$  is the old state vector

$w$  is the updated state vector

## The general case

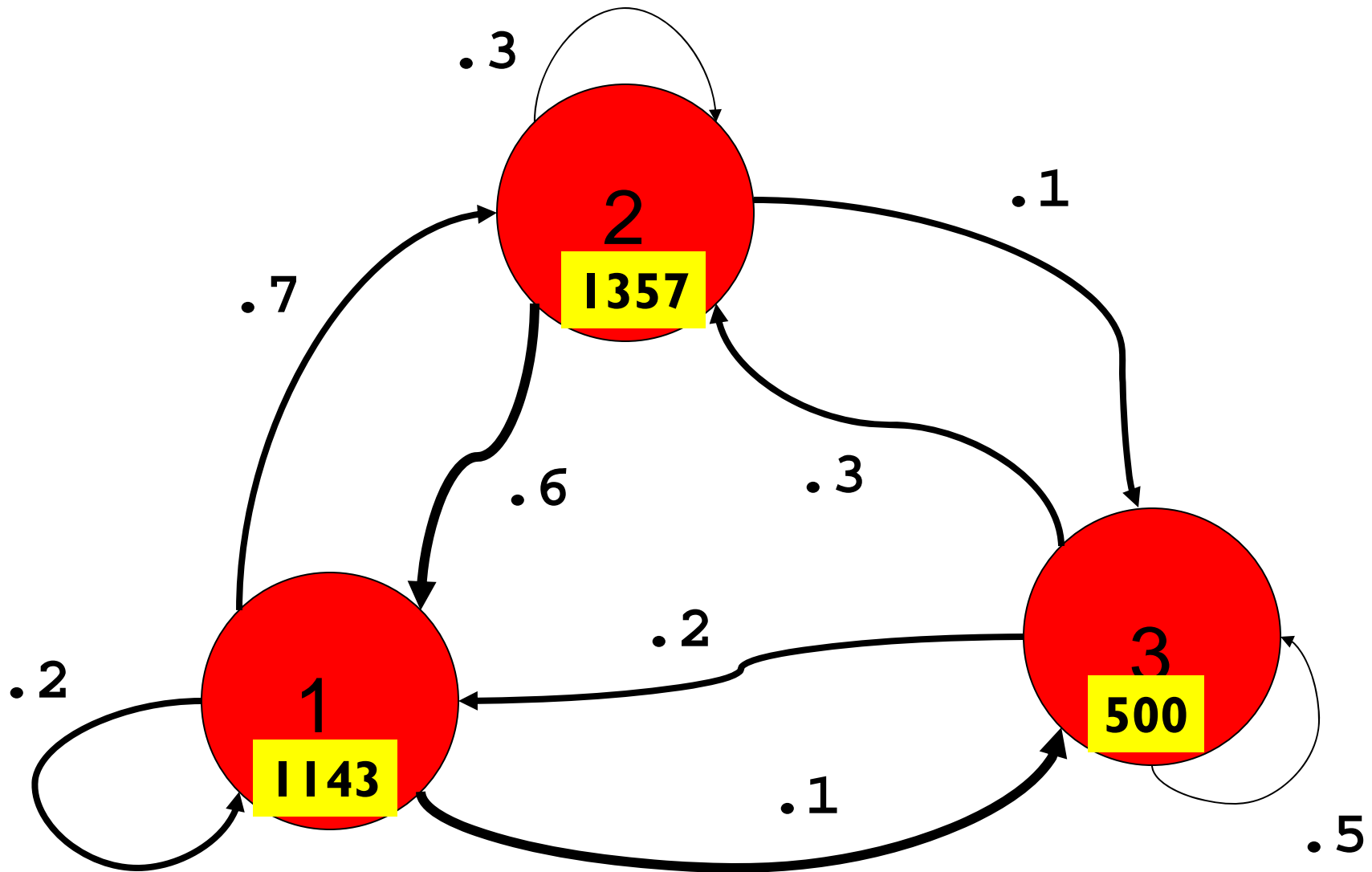
```
function w = Update(P,v)
% Update state vector v based on transition
% probability matrix P to give state vector w
n = length(v);
w = zeros(n,1);
for i=1:n
    for j=1:n
        w(i) = w(i) + P(i,j)*v(j);
    end
end
```



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
```

Stationary vector indicates importance: **2 1 3**



## A random walk on the web

Repeat:

You are on a webpage.

There are  $m$  outlinks,  
so choose one at  
random.

Click on the link.

Use the link structure of the  
web to figure out the  
transitional probabilities!

## Random island hopping

Repeat:

You are on an island.

According to the  
transitional  
probabilities,  
go to another island.

(Assume no dead ends for  
now; we deal with them later.)

# Connectivity Matrix

$G$

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

$G(i, j)$  is 1 if there is a link on page  $j$  to page  $i$ .

(i.e., you can get to  $i$  from  $j$ .)

# Connectivity Matrix

G

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 Matrix

derived from Connectivity Matrix

P

0	0	0	0	0	0	?	?
?	0	0	?	0	0	0	0
?	0	?	0	0	?	0	?
0	0	0	0	?	0	0	0
?	0	?	0	0	0	0	?
0	0	?	0	0	0	0	?
0	0	?	0	0	0	0	0
0	?	0	?	0	0	0	0

# Connectivity Matrix

**G**

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

**P**

0	0	0	0	0	0	?	?
?	0	0	?	0	0	0	0
?	0	?	0	0	?	0	?
0	0	0	0	?	0	0	0
?	0	?	0	0	0	0	?
0	0	?	0	0	0	0	?
0	0	?	0	0	0	0	0
0	?	0	?	0	0	0	0

- A. 0
- B. 1/8
- C. 1/3
- D. 1
- E. rand(I)

# Connectivity Matrix

G

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 Matrix

derived from Connectivity Matrix

P

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

## Connectivity (G) $\rightarrow$ Transition Probability (P)

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

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

The random walk idea gets the transitional probabilities from connectivity. So how to deal with dead ends?

Repeat:

You are on a webpage.

There are  $m$  outlinks.

Choose one at random.

Click on the link.

What if there are no outlinks?

The random walk idea gets transitional probabilities from connectivity. Can modify the random walk to deal 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.**

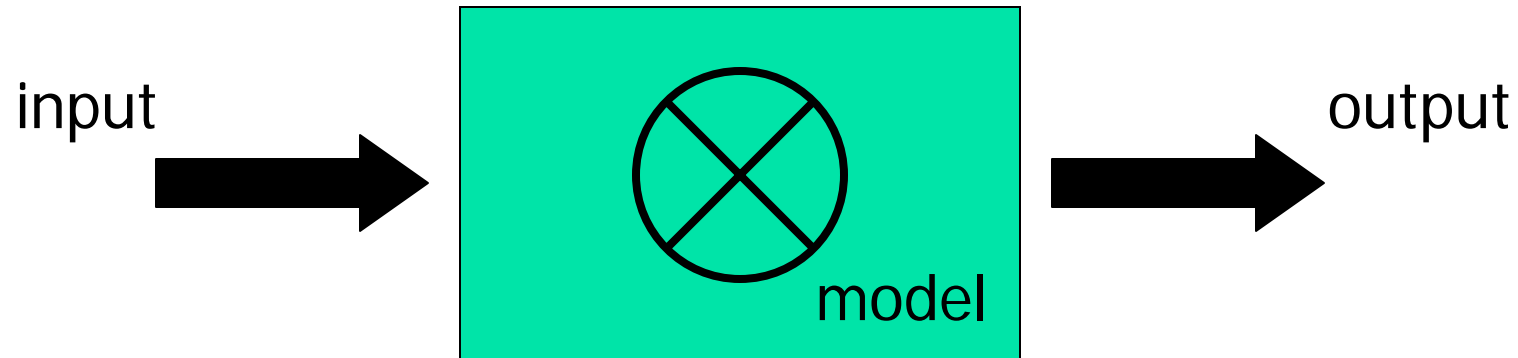
## Quantifying Importance

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”?

## Simulation vs. optimization



- **Simulation**: Specify input, run it through the model, and get result.
- **Optimization** is the reverse problem: Specify the result that you want—your objectives—and try to find what “starting values” or actions are needed

# 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?

# Enumerate all the possibilities and pick the shortest

What is the best itinerary to visit Boston, Miami, LA, Dallas?



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

$3! = 6$  possibilities

Enumerate all the possibilities and pick the shortest



What is the best itinerary to visit Boston, Miami, LA, Dallas?

$3! = 6$  possibilities

Add Seattle, NYC  
Austin, Denver

$7! = 5040$

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

Enumerate all the possibilities and pick the shortest



What is the best itinerary to visit Boston, Miami, LA, Dallas?

$3! = 6$  possibilities

Add Seattle, NYC  
Austin, Denver

$7! = 5040$

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

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

## 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”
  - Functions for reducing redundancy
  - Control flow (if-else; loops)
  - Recursion
  - Data structures
  - Graphics
  - File handling

## What we learned... (cont'd)

- Applications and concepts
  - Image and sound
  - Sorting and searching—you should know the algorithms covered
  - Divide-and-conquer strategies
  - Approximation and error
  - Simulation
  - Computational effort and efficiency



# Final Exam

- Thurs 5/12, 9-11:30am, Barton **East**
- Covers entire course; some emphasis on 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