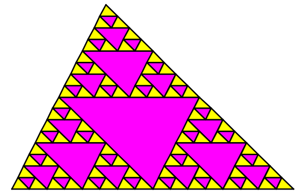
- Previous Lecture:
  - Comparing different sorting algorithms
  - Recursion
- Today's Lecture:
  - Recursion review
  - Efficiency: dealing with "expensive" function evaluation
  - A model to quantify importance: Google "Page Rank"
- Announcement:
  - Discussion this week in the computer lab (UP B7)
  - P6 due Thursday at 11pm

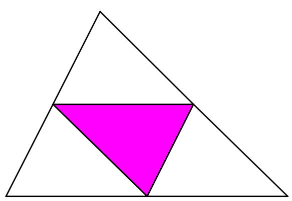
Why is mesh generation a divide-&-conquer process?

Let's draw this graphic



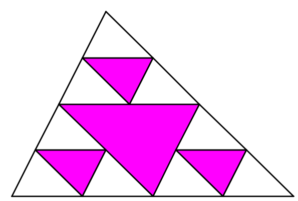
A "level-1" partition of the triangle

(obtained by connecting the midpoints of the sides of the original triangle)

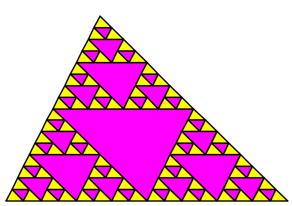


Now do the same partitioning (connecting midpts) on each corner (white) triangle to obtain the "level-2" partitioning

The "level-2" partition of the triangle



The "level-4" partition of the triangle



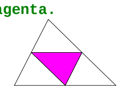
The basic operation at each level

if *the triangle is small*
    Don't subdivide and just color it yellow.
else
    Subdivide:
    Connect the side midpoints;
    color the interior triangle magenta;
    apply same process to each outer triangle.
end

```
function MeshTriangle(x,y,L)
% x,y are 3-vectors that define the vertices of a triangle.
% Draw level-L partitioning.  Assume hold is on.

if L==0
  % Recursion limit reached; no more subdivision required.
    fill(x,y,'y')  % Color this triangle yellow

else
  % Need to subdivide:  determine the side midpoints; connect
  % midpts to get "interior triangle"; color it magenta.
```

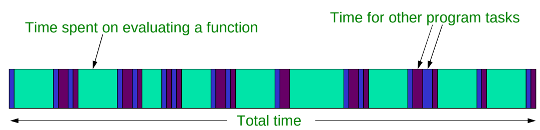


```
  % Apply the process to the three "corner" triangles...

end
```


Expensive function evaluations

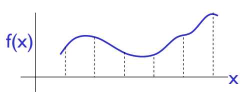
- Consider the execution of a program that is dominated by multiple calls to an expensive-to-evaluate function (e.g., climate simulation models)



- Can try to improve efficiency by dealing with the expensive function evaluations

Dealing with expensive function evaluations

- Can the function code be improved?
- Can we do fewer function evaluations?
- Can we pre-compute and store specific function values so that during the main program execution the program can just look up the values?
  - Consider function f(x). If there are many function calls and few distinct values of x, can get substantial speedup
  - Only speeds up main program execution—it still takes time to do the pre-computation



What are some issues and potential problems with the "table look-up" strategy?

x	f(x)
1	1.01
2	2.67
3	5.71
4	9.12
5	7.98
:	:

Pre-calculate and store these values (e.g., in a vector **H**)

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.)

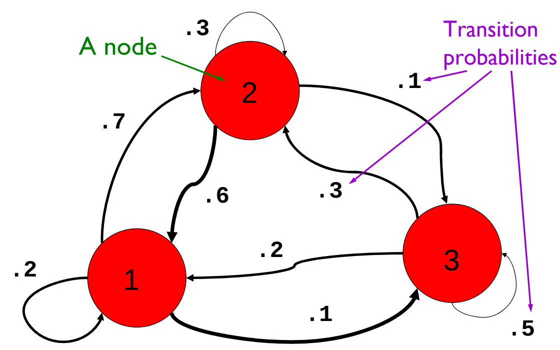
The PageRank algorithm orders these pages from "most important" to "least important."

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



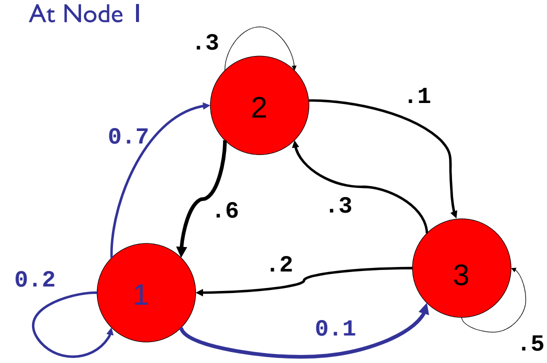
A special random walk

Suppose there are a 1000 people on each node.

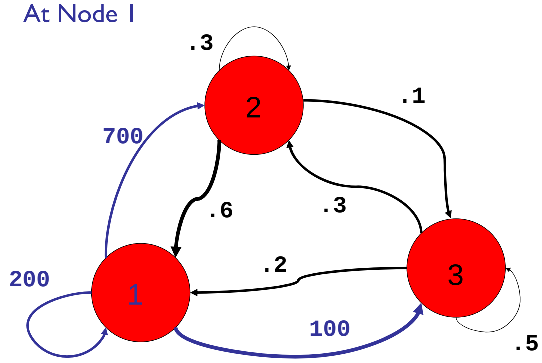
At the sound of a whistle they hop to another node in accordance with the "outbound" probabilities.

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

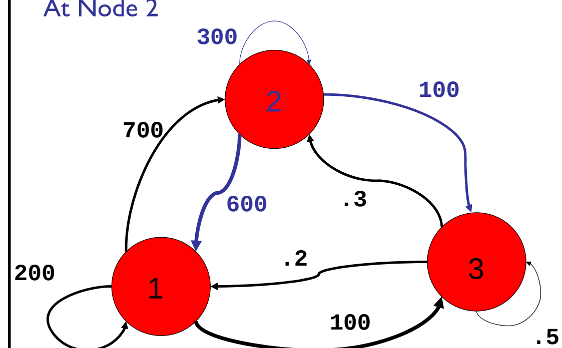
At Node 1



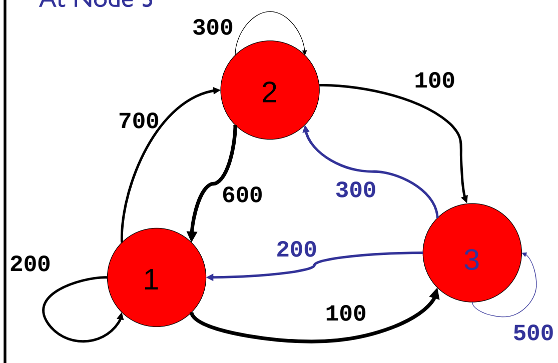
At Node 1



At Node 2



At Node 3



State Vector:
describes the state at each node at a specific time

T=0	T=1	T=2
1000	1000	1120
1000	1300	1300
1000	700	580

After 100 iterations

	T=99	T=100
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

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

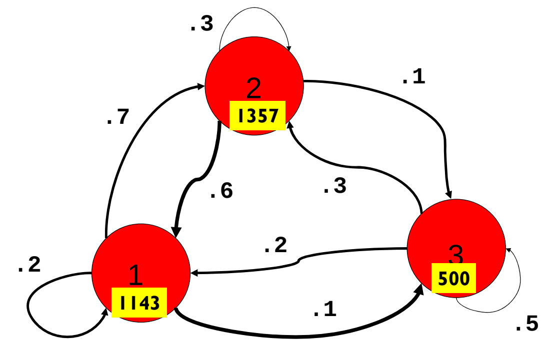
The general case

```
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



Lecture 27 59

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.

Random island hopping

Repeat:
You are on an island.
According to the transitional probabilities,
go to another island.

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

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

Lecture 27 60

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`.)

Lecture 27 61

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

Lecture 27 62

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

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

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

Lecture 27 63

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
```

Lecture 27 65

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
```

Lecture 27 66

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

Lecture 27 67

```
[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

Lecture 27 68

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?

Lecture 27 69

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.

Lecture 27 70

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"?

Lecture 27 72