- Previous Lecture:
  - Examples on vectors (1-d arrays)
- Today's Lecture:
  - 2-d array—matrix
- Announcements:
  - Project 3 due on Thursday, 10/14, at 11pm
  - No discussion on Tues (Fall Brk). Attendance at next discussion (Wednesday) is optional, but the material covered in the exercise is required. DO THE EXERCISE!
  - Prelim 2 on Thurs, Oct 21st, 7:30-9pm. Email Randy Hess if you have an exam conflict with another course. rbhess@cs.cornell.edu

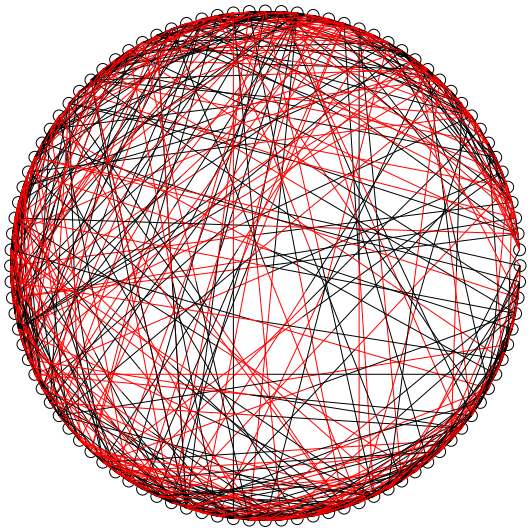
Storing and using data in *tables*

A company has 3 factories that make 5 products with these costs:

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

What is the best way to fill a given purchase order?



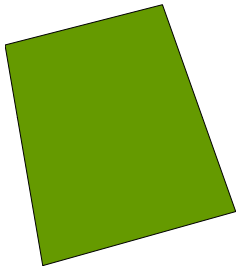
Connections between webpages

```
0 0 1 0 1 0 0
1 0 0 1 1 1 0
0 1 0 1 1 1 1
1 0 1 1 0 1 0
0 0 1 1 0 1 1
0 0 1 0 1 0 1
0 1 1 0 1 1 0
```

Color computation

- Color is a 3-vector, sometimes called the RGB values
- Any color is a mix of red, green, and blue
- Example:

```
c= [0.4  0.6  0]
```



- Each component is a real value in [0,1]
- [0 0 0] is black
- [1 1 1] is white

Start with drawing a single line segment

```
a= 0;   % x-coord of pt 1
b= 1;   % y-coord of pt 1
c= 5;   % x-coord of pt 2
d= 3;   % y-coord of pt 2
plot([a c], [b d], '-*')
```

Line/marker format

x-values (a vector)

y-values (a vector)

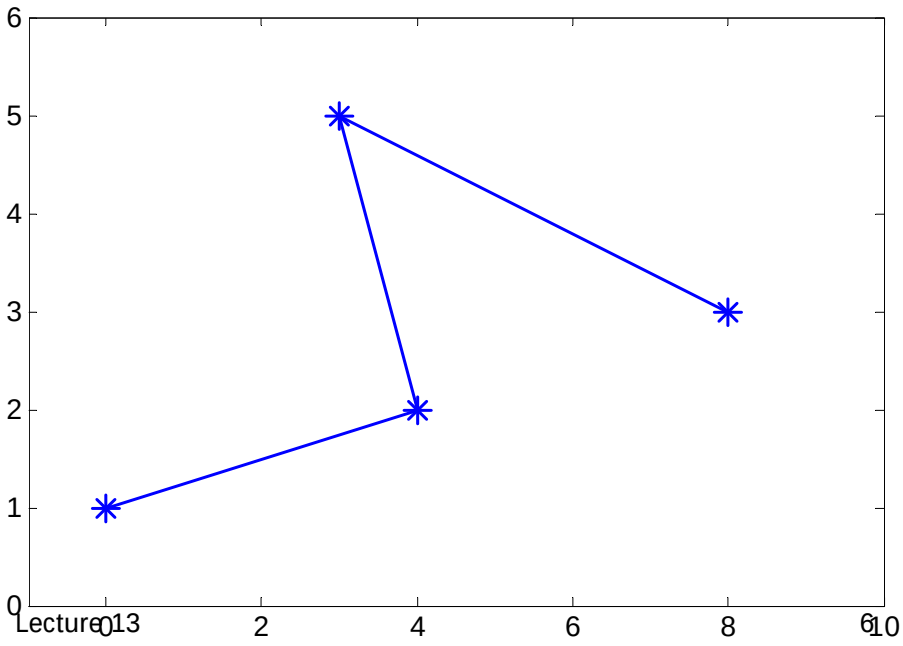
Making an x-y plot

```
a= [0 4 3 8];   % x-coords
b= [1 2 5 3];   % y-coords
plot(a, b, '-*')
```

x-values
(a vector)

y-values
(a vector)

Line/marker
format



Drawing a polygon (multiple line segments)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h.
x= [                                ];  % x data
y= [                                ];  % y data
plot(x, y)
```

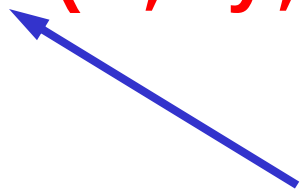
Fill in the missing vector values!

Drawing a polygon (multiple line segments)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h.
x= [a  a+w  a+w  a    a ];  % x data
y= [b  b    b+h  b+h  b ];  % y data
plot(x, y)
```

Coloring a polygon (fill)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h,
% and fill it with a color named by c.
x= [a  a+w  a+w  a    a];  % x data
y= [b  b    b+h  b+h  b];  % y data
fill(x, y, c)
```



A built-in function

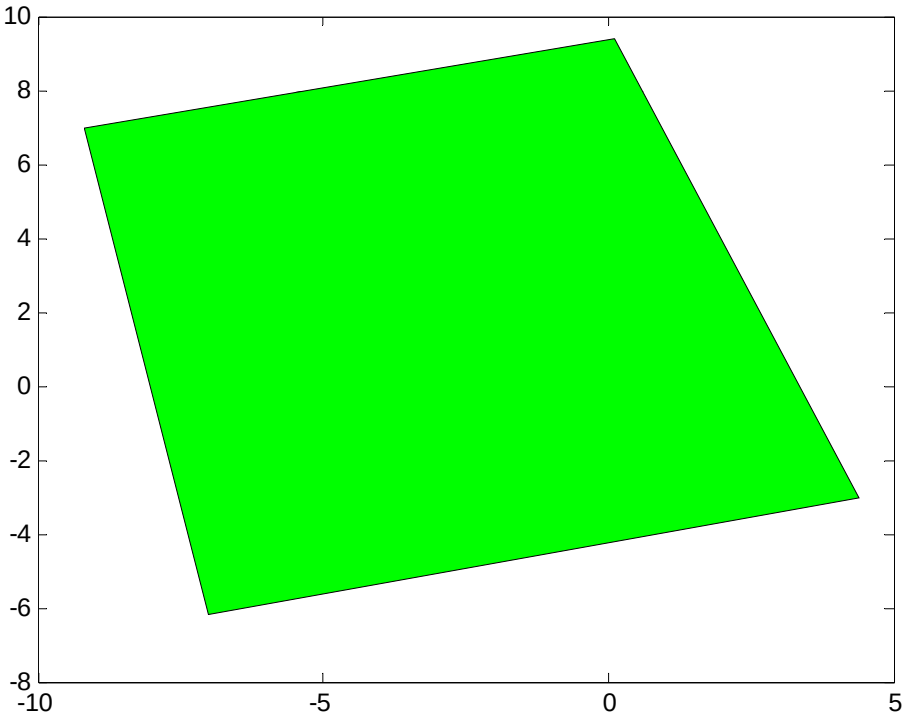
Coloring a polygon (fill)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h,
% and fill it with a color named by c.
x= [a   a+w   a+w   a     a];   % x data
y= [b   b     b+h   b+h   b];   % y data
fill(x, y, c)
```

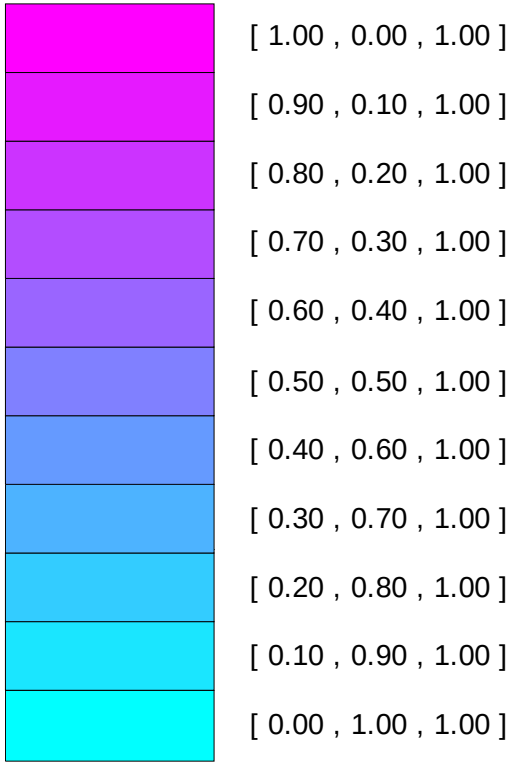
Built-in function **fill** actually does the "wrap-around" automatically.

```
x= [0.1 -9.2  -7  4.4];
y= [9.4  7   -6.2  -3];
fill(x,y,'g')
```

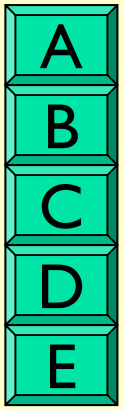
Can be a vector
(RGB values)



```
function paintChips(c1,c2,n)
% n tiles from color c1 to c2



for k= 0:n-1
    % Compute color of kth tile
    f= ???
    v= (1-f)*c1 + f*c2;
    % Draw kth tile



end
```

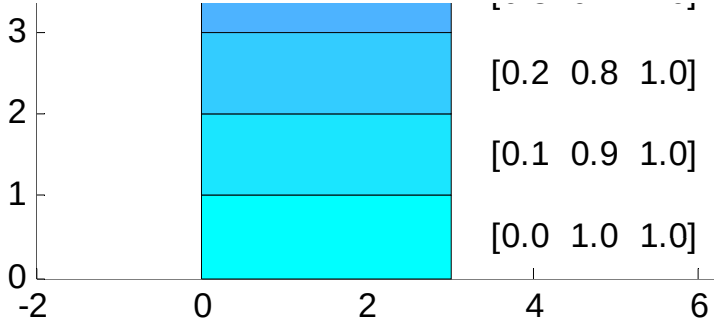


```
function paintChips(c1,c2,n)
% n tiles from color c1 to c2



for k= 0:n-1
    % Compute color of kth tile
    f= ???
    v= (1-f)*c1 + f*c2;
    % Draw kth tile




end
```

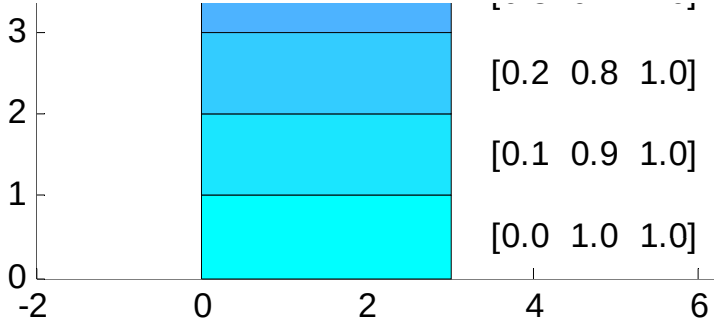


k/n
k/(n-1)
(k-1)/n
(k-1)/(n-1)
(k-1)/(n+1)

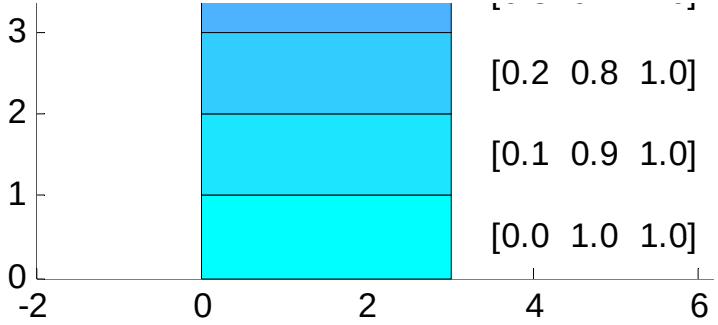
```
function paintChips(c1,c2,n)
% n tiles from color c1 to c2




for k= 0:n-1
    % Compute color of kth tile
    f= ???
    v= (1-f)*c1 + f*c2;
    % Draw kth tile




end
```



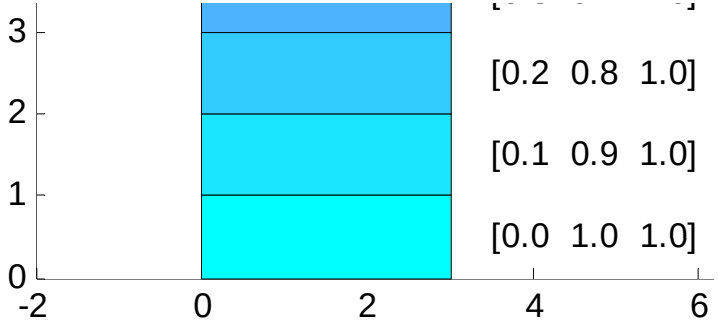
```
function paintChips(c1,c2,n)
% n tiles from color c1 to c2

x= [0 3 3 0];
y= [0 0 1 1];
for k= 0:n-1
    % Compute color of kth tile
    f= k/(n-1);
    v= (1-f)*c1 + f*c2;
    % Draw kth tile
    fill(_____, _____, v)



end
```



```
function paintChips(c1,c2,n)
% n tiles from color c1 to c2

x= [0 3 3 0];
y= [0 0 1 1];
for k= 0:n-1
    % Compute color of kth tile
    f= ???
    v= (1-f)*c1 + f*c2;
    % Draw kth tile
    fill(_____, _____, v)



end
```



```
function paintChips(c1,c2,n)
% n tiles from color c1 to c2

x= [0 3 3 0];
y= [0 0 1 1];
for k= 0:n-1
    % Compute color of kth tile
    f= k/(n-1);
    v= (1-f)*c1 + f*c2;
    % Draw kth tile
    fill(x, y+k, v)
    text(3.5, k+.5, ...
         sprintf('[%.1f %.1f %.1f]',...
         v(1),v(2),v(3) )
end
```



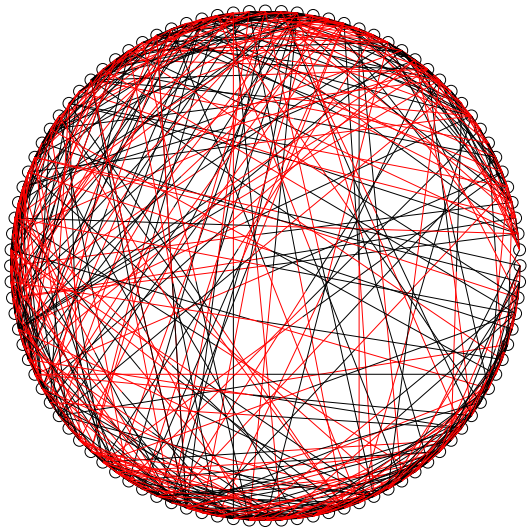
Storing and using data in *tables*

A company has 3 factories that make 5 products with these costs:

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

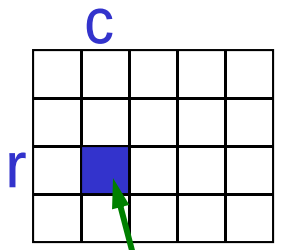
What is the best way to fill a given purchase order?



Connections between webpages

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

2-d array: **matrix**



- An array is a named collection of like data organized into rows and columns
- A 2-d array is a table, called a *matrix*
- Two *indices* identify the position of a value in a matrix, e.g.,

  `mat(r,c)`

  refers to component in row r, column c of matrix mat
- Array index starts at 1
- Rectangular: all rows have the same #of columns

Creating a matrix

- Built-in functions: ones, zeros, rand
  - E.g., zeros(2,3) gives a 2-by-3 matrix of 0s
- "Build" a matrix using square brackets, [ ], but the dimension must match up:
  - [x y] puts y to the right of x
  - [x; y] puts y below x
  - [4 0 3; 5 1 9] creates the matrix

4	0	3
5	1	9

  - [4 0 3; ones(1,3)] gives

4	0	3
1	1	1

  - [4 0 3; ones(3,1)] doesn't work

Working with a matrix:
`size` and individual components

Given a matrix M

2	-1	.5	0	-3
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

```
[nr, nc]= size(M)  % nr is #of rows,
                   % nc is #of columns
nr= size(M, 1)  % # of rows
nc= size(M, 2)  % # of columns

M(2,4)= 1;
disp(M(3,1))
M(1,nc)= 4;
```

```
% What will M be?
M = [ones(1,3); 1:4]
```

A

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

B

```
1   1   1
1   2   3
```

C

Error – M not created

What will **A** be?

```
A= [1  1]
A= [A'  ones(2,1)]
A= [1  1  1  1;  A  A]
```

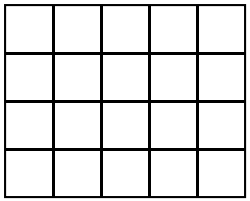
A 3-by-4 matrix

B 4-by-3 matrix

C vector of length 12

D *Error*

Example: minimum value in a matrix



```
function val = minInMatrix(M)
% val is the smallest value in matrix M
```

minInMatrix.m

Pattern for traversing a matrix M

```
[nr, nc] = size(M)
for  r= 1:nr
      % At row r
      for  c= 1:nc
            % At column c (in row r)
            %
            % Do something with M(r,c) …
      end
end
```

Matrix example: Random Web

- N web pages can be represented by an N-by-N Link Array A.
- A(i,j) is 1 if there is a link on webpage j to webpage i
- Generate a random link array and display the connectivity:
  - There is no link from a page to itself
  - If i≠j then A(i,j) = 1 with probability $\frac{1}{1+|i-j|}$

  ⇨ There is more likely to be a link if i is close to j

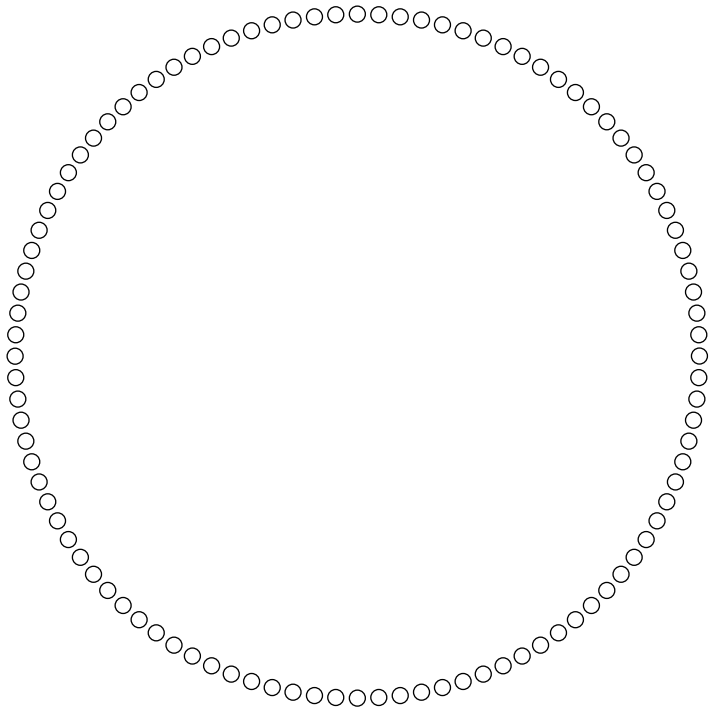
```
function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages

A = zeros(n,n);
for i=1:n
  for j=1:n
    r = rand(1);
    if i~=j && r<= 1/(1 + abs(i-j));
        A(i,j) = 1;
    end
  end
end
```

Random web

N = 20

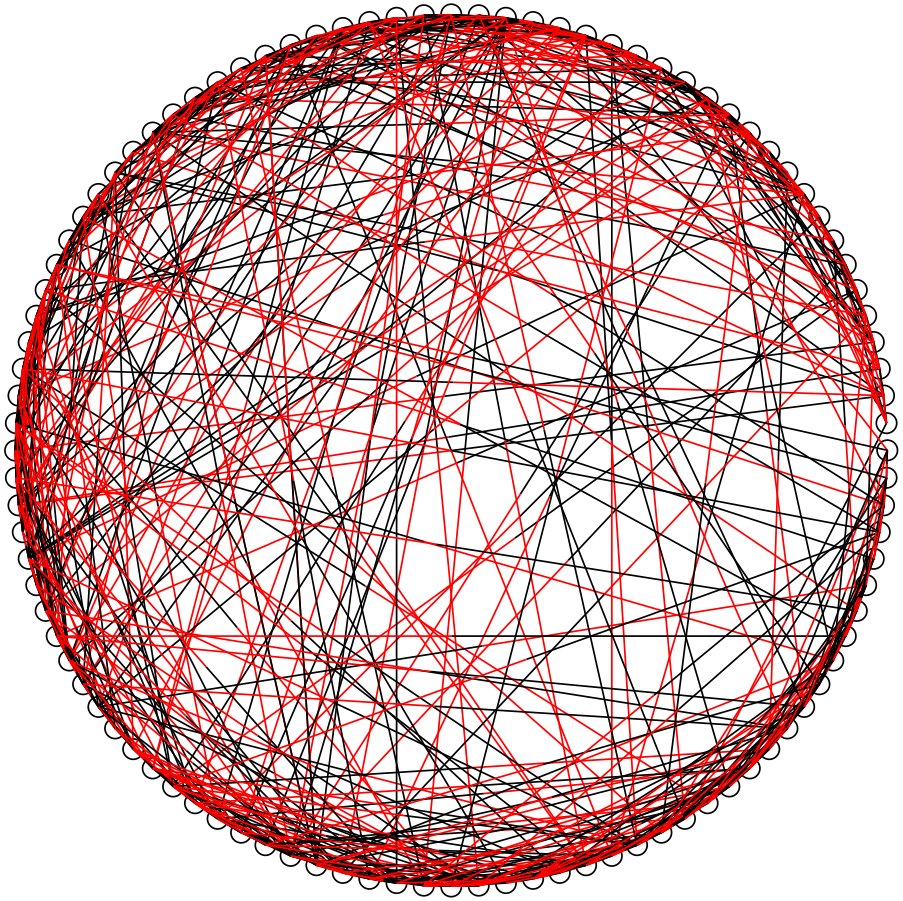
```
0 1 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0
0 1 1 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1
0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0 1 1 0 1 0 1 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 1
0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0
0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1
0 0 0 0 0 0 1 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 1 0 1 0 0
```

Represent the web pages graphically…



100 Web pages arranged in a circle.
Next display the links….

Represent the web pages graphically…



Line black as it leaves page j, red when it arrives at page i

ShowRandomLinks.m