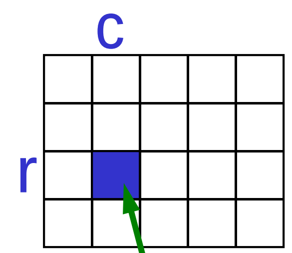
- Today's Lecture:
  - 2-d array—matrix
  - Function & subfunction
  - Details on `for`-loop (see blecture)

- Announcements:
  - Friday: lab session in Upson B7
  - Assignment 1b due Tuesday 11:59pm
  - Test 1 on Thursday in class; review on Tuesday.

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

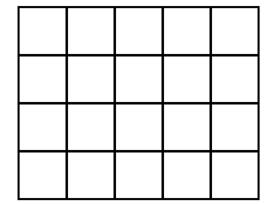
Error – M not created

What will **A** be?

```
A= [0  0]
A= [A’  ones(2,1)]
A= [0  0  0  0;  A  A]
```

```
% Given an nr-by-nc matrix M.
% What is A?
for  r= 1: nr
    for c= 1: nc
        A(c,r)= M(r,c);
    end
end
```

Example: minimum value in a matrix



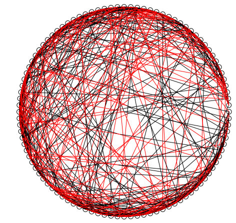
```
function val = minInMatrix(M)
% val is the smallest value in matrix M
```

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 $i \neq j$ 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

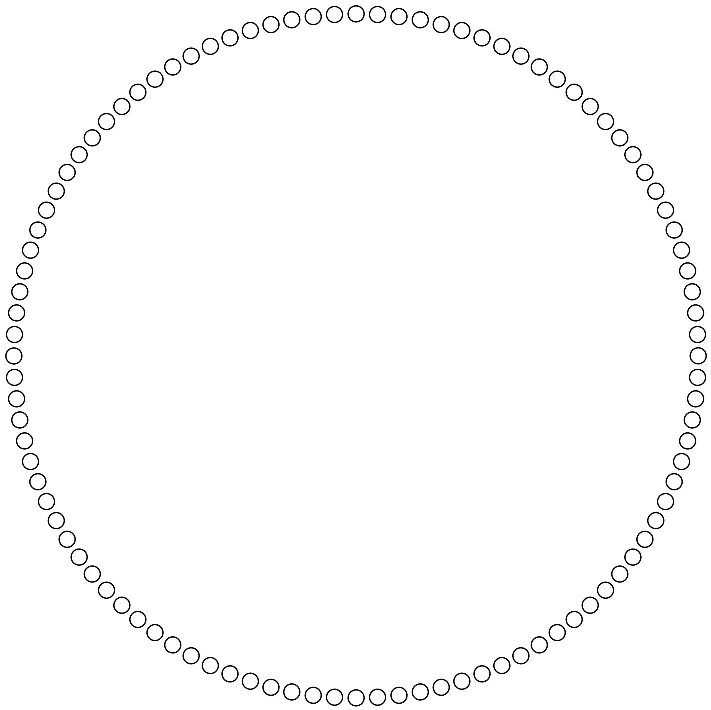
```
function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages

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

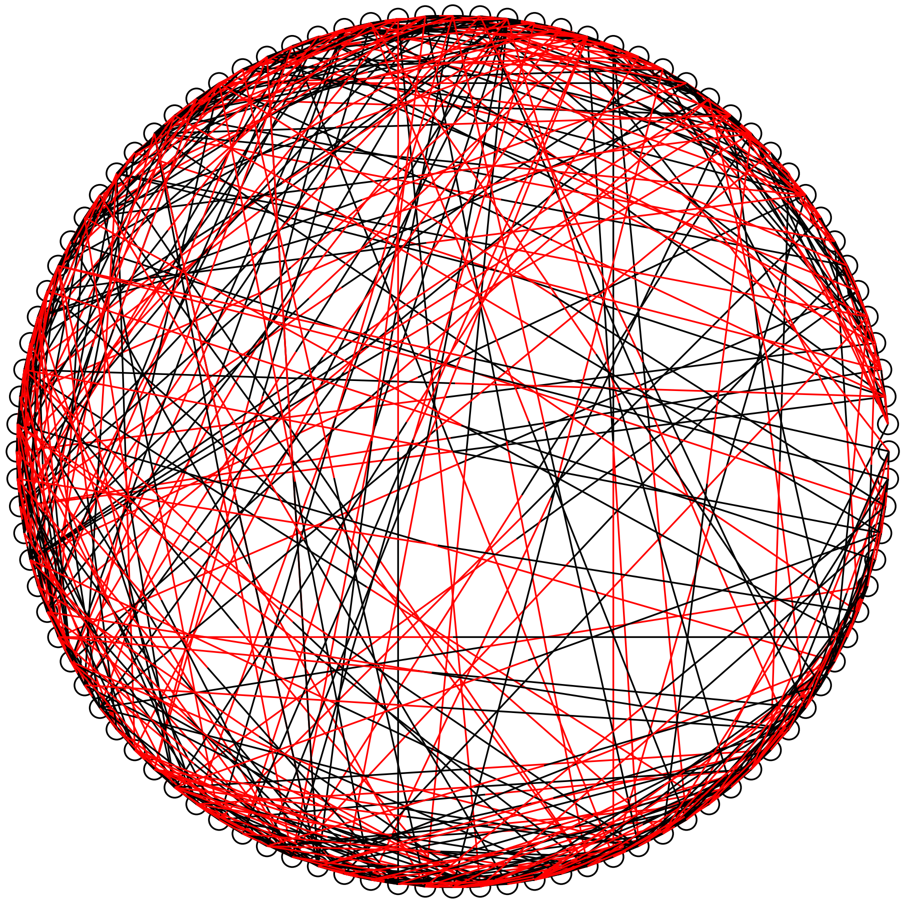
0	1	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
0	1	1	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0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

Local minimum in a neighborhood

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

Component (2,3)

Neighborhood of component (2,3)

Accessing a submatrix

M

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

Component (2,3)

`M(2,3)`

Neighborhood of component (2,3)

`M(1:3,2:4)`

Local minimum in a neighborhood

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

Component (3,5)

Neighborhood of component (3,5)

Local minimum in a neighborhood

- Write a function minInNeighborhood
- Input parameters:
  - M: matrix of numeric values
  - loc: location of the middle of the neighborhood
    loc(1), loc(2) are the row, column numbers
- Output parameter: minVal
  The minimum value of the neighborhood

Ask yourself questions!

- Can you find the min of a (sub)matrix?
  - Yes! Our function minInMatrix(A)
- Given the indices r, c (representing element M(r,c)), is it easy to define the neighborhood?
  - Yes, for the general case the neighborhood is

    M(r-1:r+1, c-1:c+1)
  - But need to deal with the "border cases"

Local minimum in a neighborhood

M

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

Component (3,5)

Want to be able to use the general case,

`M(r-1:r+1,c-1:c+1)`

Local minimum in a neighborhood

M

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

Want to be able to use the general case,

`m(r-1:r+1,c-1:c+1)`

Local minimum in a neighborhood

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

Create a border of values (B is some big number)

Want to be able to use the general case,

`m(r-1:r+1,c-1:c+1)`

Note: This is an exercise on manipulating a matrix. Method not suitable for a large matrix!

minInNeighborhood.m

minInNeighborhoodV2.m

minInNeighborhoodV3.m

Subfunction

- There can be more than one function in an M-file
- top function is the main function and has the name of the file
- remaining functions are subfunctions, accessible only by the functions in the same m-file
- Each (sub)function in the file begins with a function header
- Keyword **end** is not necessary at the end of a (sub)function