

- Previous Lecture:

- Discrete vs. continuous; finite vs. infinite
- Linear interpolation
- Vectorized operations

- Today's Lecture:

- 2-d array—matrix

- Announcements:

- Discussion this week in the classrooms as listed in the roster
- Prelim I tonight at 7:30pm
 - Last names A-O: Uris Auditorium (room G01)
 - Last names P-Z: Upson Auditorium (room B17)

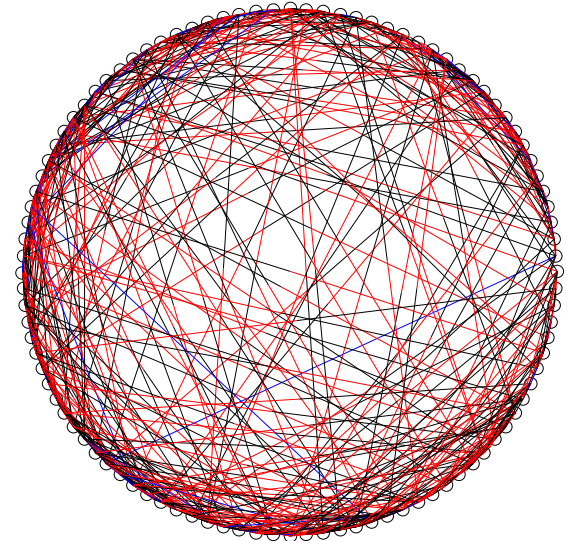
Storing and using data in tables

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

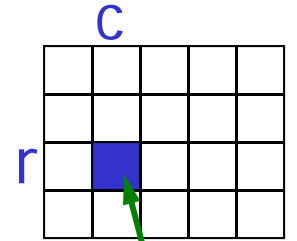
What is the best way to fill a given purchase order?



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**



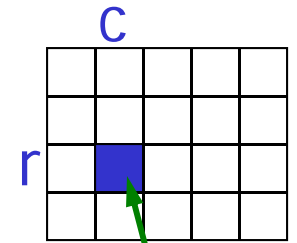
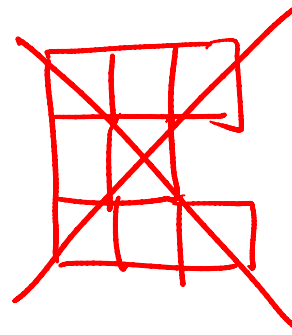
- An array is a **named** collection of **like** data organized into rows and columns
- A 2-d array is a table, called a **matrix**
- Two **indices** identify the position of a value in a matrix, e.g.,

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

2-d array: **matrix**



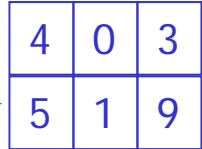
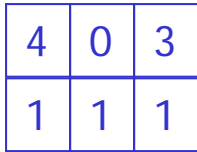
- An array is a **named** collection of **like** data organized into rows and columns
- A 2-d array is a table, called a **matrix**
- Two **indices** identify the position of a value in a matrix, e.g.,

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; ones(1,3)]` gives 
 - `[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;
```

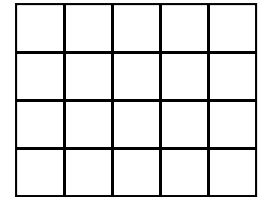
```
disp(M(3,1))
```

```
M(1,nc)= 4;
```

Example: minimum value in a matrix

```
function val = minInMatrix(M)
```

% val is the smallest value in matrix M



Example: minimum value in a matrix

```
function val = minInMatrix(M)
```

% val is the smallest value in matrix M

```
[nr, nc] = size(M);
```

```
val = M(1,1);
```

```
for r = 1:nr
```

```
    % At row r
```

```
    for c = 1:nc
```

```
        % At col c (at row r)
```

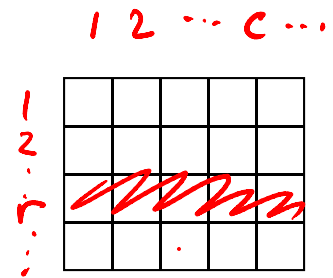
```
        if M(r,c) < val
```

```
            val = M(r,c);
```

```
        end
```

```
    end
```

```
end
```



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
end
```

% 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
```

 A is M with the columns in reverse order

 A is M with the rows in reverse order

 A is the transpose of M

 A and M are the same

% 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
```

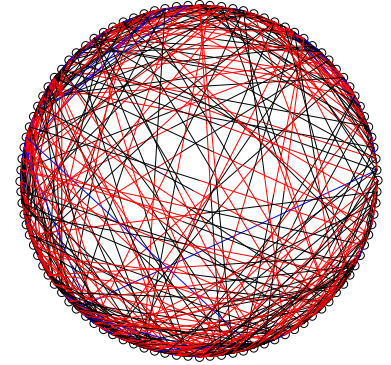
M

0	3	2	5
4	13	20	6
11	26	9	1

A

0	4	11
3	13	26
2	20	9
5	6	1

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

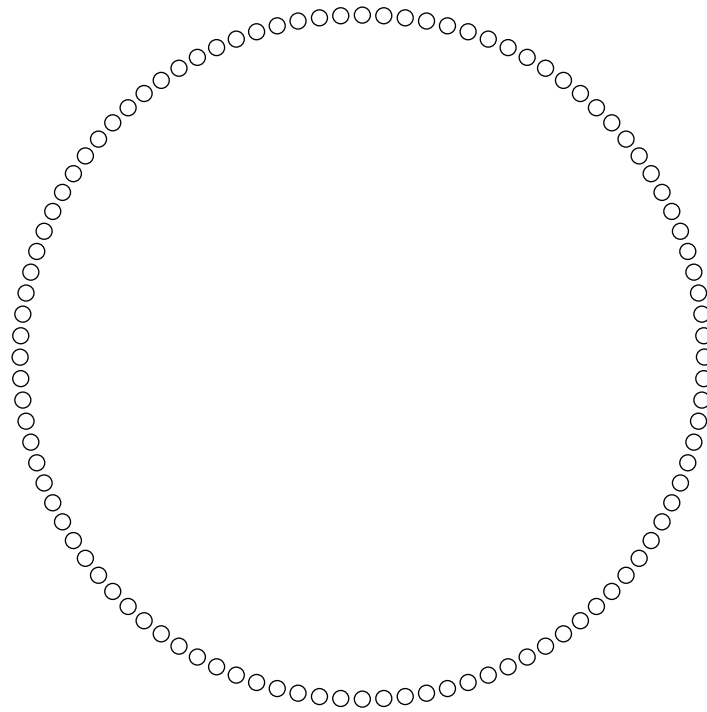
```
function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages

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
end
```

Random web
N = 20

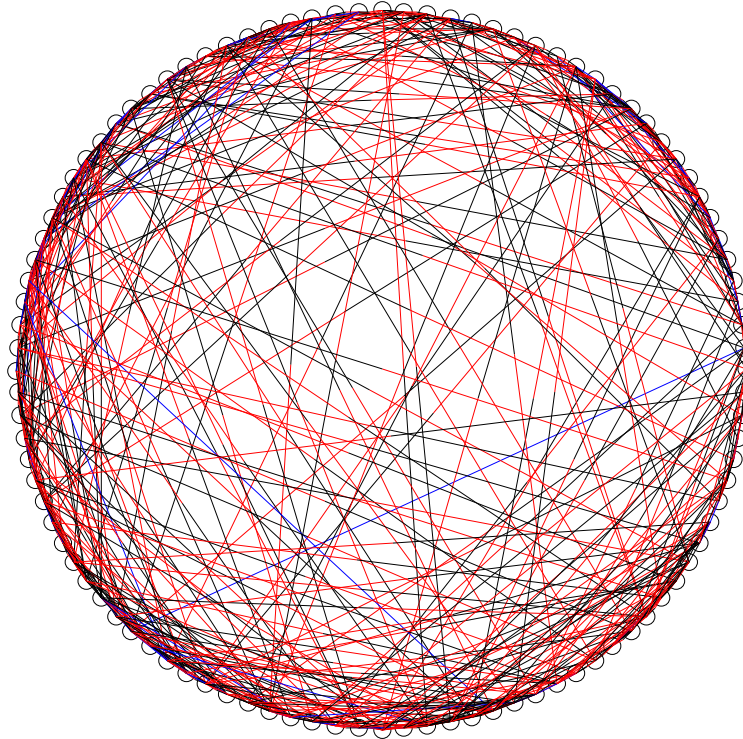
```
0 1 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
0 1 1 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0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

Represent the web pages graphically...



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

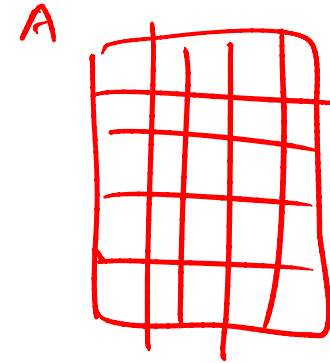
Represent the web pages graphically...



Bidirectional links are blue. Unidirectional link is black as it leaves page j, red when it arrives at page i.


```
for i = 1:n  
    for j = 1:n
```

```
    end  
end
```



Is there another way? See
ShowRandomLinks.m

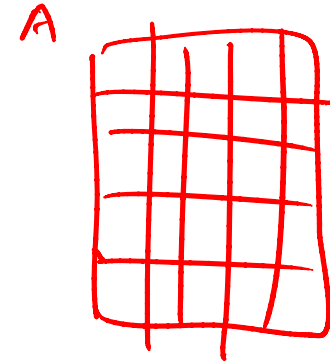
```
for i = 1:n
```

```
    for j = 1:n
```

```
        if A(i,j) == 1 && A(j,i) == 1
            % Blue
```

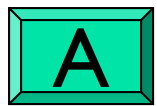
```
        elseif A(i,j) == 1
            % Black-Red
            j → mid    mid → i
```

```
        end
    end
end
```

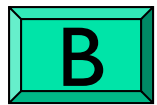


Somewhat inefficient: each blue line gets drawn twice.
See **ShowRandomLinks.m**

```
% Given an n-by-m matrix A.  
% What is this operation?  
for g= 1: n  
    for h= 1: floor(m/2)  
        A(g,h)= A(g, m-h+1);  
    end  
end
```



Reflect the right half of **A**
onto the left half



Reflect the bottom half of
A onto the top half

```
% Given an nr-by-nc matrix A.  
% What is this operation?  
for r= 1: nr  
    for c= 1: floor(nc/2)  
        A(r,c)= A(r, nc-c+1);  
    end  
end
```

- a. Reflect the right half of **A** onto the left half
- b. Reflect the bottom half of **A** onto the top half