- 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

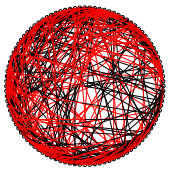
Storing and using data in *tables*

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
```


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

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

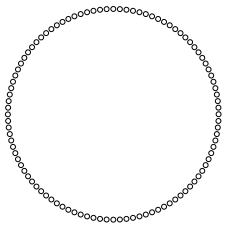

```
function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages

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

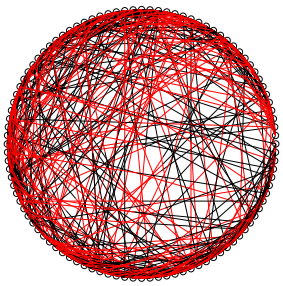
```
01110000010010000000
10001000111000000100
01010000000000000000
00101000000000000000
00010000001100000000
00000000000001010000
01111100010110000000
00000010000100000011
01000000010010001000
00000001101000000001
00000010000011000000
00000010010000000001
00010000110101100000
00000010000000110000
00000101000010010001
00000010001000001010
01000000100001010110
00000000000000011001
00000010000000000000
00000000000000001010
```

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

A Cost/Inventory Problem

- A company has 3 factories that make 5 different products
- The cost of making a product varies from factory to factory
- The inventory/capacity varies from factory to factory

Problems

A customer submits a purchase order that is to be filled by a single factory.

1. How much would it cost a factory to fill the order?
2. Does a factory have enough inventory/capacity to fill the order?
3. Among the factories that can fill the order, who can do it most cheaply?

Cost Array

`C`

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

The value of `C(i,j)` is what it costs factory i to make product j.

Inventory (or Capacity) Array

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

The value of `Inv(i,j)` is the inventory in factory i of product j.

Purchase Order

PO

1	0	12	29	5

The value of `PO(j)` is the number of product j's that the customer wants

C

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

PO

1	0	12	29	5

Cost for factory 2:

```
s = 0;  %Sum of cost
for j=1:5
  s = s + C(2,j)*PO(j)
end
```


C

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

PO

1	0	12	29	5

Cost for factory i:

```
s = 0;  %Sum of cost
for j=1:5
  s = s + C(i,j)*PO(j)
end
```


Encapsulate…

```
function  TheBill = iCost(i,C,PO)
% The cost when factory i fills the
% purchase order

nProd = length(PO);
TheBill = 0;
for j=1:nProd
   TheBill = TheBill + C(i,j)*PO(j);
end
```


Finding the Cheapest

C

10	36	22	15	62	1019
12	35	20	12	66	930
13	37	21	16	59	1040

PO

1	0	12	29	5

As computed by `iCost`

Finding the Cheapest

```
iBest = 0;  minBill = inf;
for i=1:nFact
   iBill = iCost(i,C,PO);
   if iBill < minBill
      % Found an Improvement
      iBest = i; minBill = iBill;
   end
end
```


`inf` – a special value that can be regarded as positive infinity

```
x = 10/0    assigns inf to x
y = 1+x     assigns inf to y
z = 1/x     assigns 0 to z
w < inf     is always true if w is numeric
```


Inventory/Capacity Considerations

What if a factory lacks the inventory/capacity to fill the purchase order?

Such a factory should be excluded from the find-the-cheapest computation.

Who Can Fill the Order?

Inv	38	5	99	34	42	Yes
	82	19	83	12	42	No
	51	29	21	56	87	Yes
PO	1	0	12	29	5	

Wanted: A True/False Function

DO is "true" if factory i can fill the order.

DO is "false" if factory i cannot fill the order.

Encapsulate…

```
function  DO = iCanDo(i,Inv,PO)
% DO is true if factory i can fill
% the purchase order. Otherwise, false

nProd = length(PO);
DO = 1;
for j = 1:nProd
     DO = DO && ( Inv(i,j) >= PO(j) );
end
```

Encapsulate…

```
function  DO = iCanDo(i,Inv,PO)
% DO is true if factory i can fill
% the purchase order. Otherwise, false
nProd = length(PO);
j = 1;
while j<=nProd && Inv(i,j)>=PO(j)
   j = j+1;
end
DO = _______;
```

DO should be true when…

A `j < nProd`

B `j == nProd`

C `j > nProd`

Back To Finding the Cheapest

```
iBest = 0; minBill = inf;
for i=1:nFact
   if iCanDo(i,Inv,PO)
      iBill = iCost(i,C,PO);
      if iBill < minBill
      % Found an Improvement
         iBest = i; minBill = iBill;
      end
   end
end
```


Cheapest.m

Finding the Cheapest

C	10	36	22	15	62	1019	Yes
	12	35	20	12	66	930	No
	13	37	21	16	59	1040	Yes
PO	1	0	12	29	5	⬆	⬆

As computed by `iCost` *As computed by* `iCanDo`