

- Previous Lecture:
 - 2-d array—matrix

- Today's Lecture:
 - More examples on matrices
 - Optional reading: contour plot (7.2, 7.3 in *Insight*)

- Announcement:
 - Fall Break next Mon & Tues: no lec, dis, office/consulting hrs. Attendance at 10/16 (W) dis is optional, but the exercise is required. Attend any of the 10/16 dis sections if you wish. Location is Hollister 401.
 - *Optional review sessions*: W 5:15-6:45p, W 7:30-9p.
Locations TBA

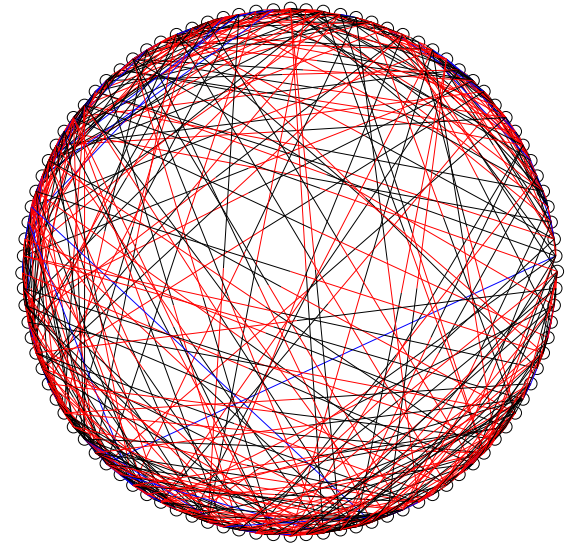
Storing and using data in tables

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?



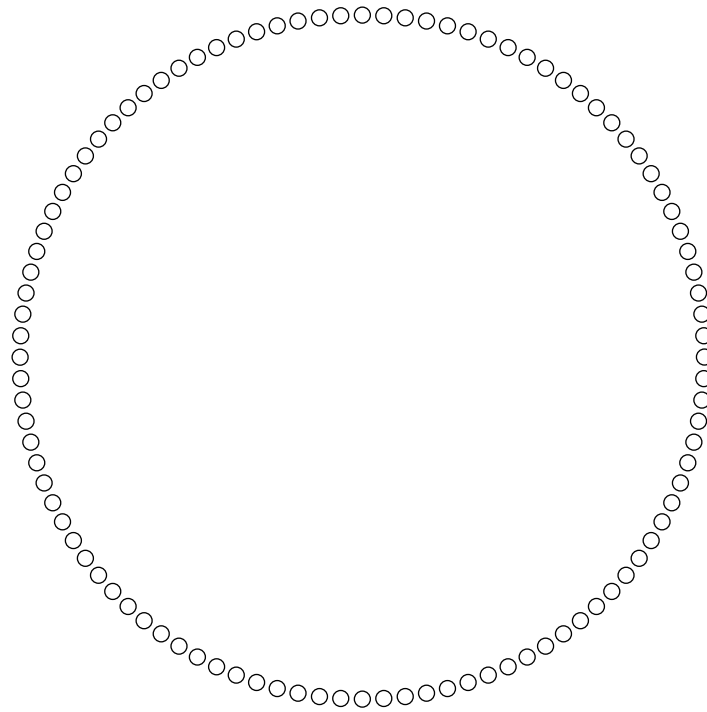
Connections
between webpages

| | | | | | | |
|---|---|---|---|---|---|---|
| 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 |

Pattern for traversing a matrix M

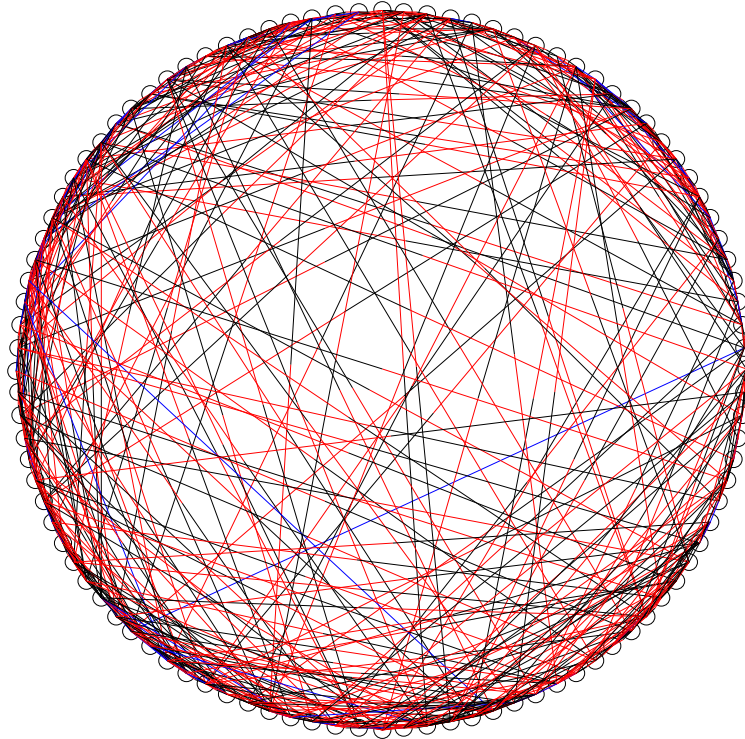
```
[nr, nc] = size(M)
for r= 1:nr
    % At row r
    for c= 1:nc
        % At column c (in row r)
        %
        % Do something with M(r,c) ...
    end
end
end
```

Represent the web pages graphically...



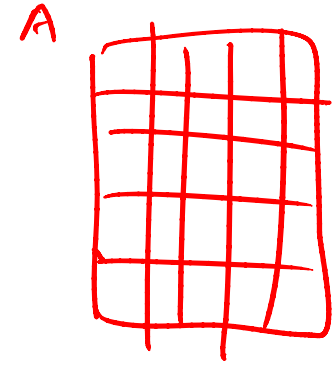
100 Web pages arranged in a circle.
Next display the links....

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
```

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

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

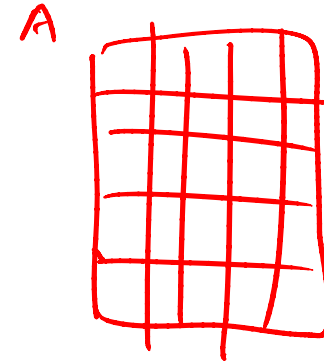
```
    if A(i,j) == 1 && A(j,i) == 1  
      % Blue
```

```
    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`

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 $\text{Inv}(i, j)$ 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 I:

$$1*10 + 0*36 + 12*22 + 29*15 + 5*62$$

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 1:

```
s = 0; %Sum of cost
for j=1:5
    s = s + C(1,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 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

```
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.

Example: Check inventory of factory 2

| | | | | | |
|-----|----|----|----|----|----|
| Inv | 38 | 5 | 99 | 34 | 42 |
| | 82 | 19 | 83 | 12 | 42 |
| | 51 | 29 | 21 | 56 | 87 |
| PO | 1 | 0 | 12 | 29 | 5 |

Method 1: check the inventory for every product

Initialization

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

DO 1

| | | | | | |
|----|---|---|----|----|---|
| PO | 1 | 0 | 12 | 29 | 5 |
|----|---|---|----|----|---|

Still True...

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

DO 1

| | | | | | |
|----|---|---|----|----|---|
| PO | 1 | 0 | 12 | 29 | 5 |
|----|---|---|----|----|---|

`DO = DO && (Inv(2,1) >= PO(1))`

Still True...

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

DO 1

| | | | | | |
|----|---|---|----|----|---|
| PO | 1 | 0 | 12 | 29 | 5 |
|----|---|---|----|----|---|

`DO = DO && (Inv(2,2) >= PO(2))`

Still True...

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

DO 1

| | | | | | |
|-----------|---|---|----|----|---|
| PO | 1 | 0 | 12 | 29 | 5 |
|-----------|---|---|----|----|---|

DO = DO && (Inv(2,3) >= PO(3))

No Longer True...

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

DO 0

| | | | | | |
|----|---|---|----|----|---|
| PO | 1 | 0 | 12 | 29 | 5 |
|----|---|---|----|----|---|

DO = DO && (Inv(2,4) >= PO(4))

Stay False...

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

DO 0

| | | | | | |
|----|---|---|----|----|---|
| PO | 1 | 0 | 12 | 29 | 5 |
|----|---|---|----|----|---|

DO = DO && (Inv(2,5) >= PO(5))

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
```

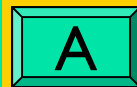
Encapsulate...

```
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 = _____;
```

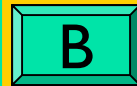
Encapsulate...

```
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...



$j < nProd$



$j == nProd$



$j > nProd$

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 = (j>nProd);
```

Back To 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
```

Don't bother with this unless there is sufficient inventory.

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
```

See `Cheapest.m`
for alternative implementation

Finding the Cheapest

