- **Previous Lecture:**
  - Vectorized code
  - 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 10/12 (W) discussion is **optional**, but do the exercise even if you don’t attend.  Attend any of the 10/12 dis sections if you wish.  Location is Hollister 464
  - **Optional review sessions:**  W7:30-9p, W5-6:30p.  Location:  Olin 255
Storing and using data in *tables*

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

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>36</th>
<th>22</th>
<th>15</th>
<th>62</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>35</td>
<td>20</td>
<td>12</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>37</td>
<td>21</td>
<td>16</td>
<td>59</td>
<td></td>
</tr>
</tbody>
</table>

What is the best way to fill a given purchase order?
Pattern for traversing a matrix $M$

$$\begin{bmatrix} \text{nr}, \text{nc} \end{bmatrix} = \text{size}(M)$$

for $r = 1:\text{nr}$
  % At row $r$
  for $c = 1:\text{nc}$
    % 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 $j$. 

- $A(i,j)$ is 1 if there is a link on webpage $j$ to webpage $i$
An event happens with probability $p$, $0 < p < 1$

% Flip a fair coin
r = rand;
if $r \leq 0.5$
    disp('heads')
else
    disp('tails')
end

% Unfair coin: shows heads twice as often as tails
r = rand;
if $r \leq \frac{2}{3}$
    disp('heads')
else
    disp('tails')
end

% Event X happens with probability $p$
r = rand;
if $r \leq p$
    % Code for event X
end
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;
            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
1 0 0 0 1 0 0 0 1 1 1 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 1 0 1 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 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 1 1 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 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 1 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 0
0 0 0 0 1 0 0 0 0 1 1 0 1 0 1 1 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 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 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0
1 0 0 0 0 0 1 0 0 0 1 0 1 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 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 1 0 0 1
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
% 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
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 \rightarrow \text{mid} \quad \text{mid} \rightarrow i
    end
end
end

Somewhat inefficient: each blue line gets drawn twice.
See ShowRandomLinks.m
Transpose—like switching row and column indices
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?
Available data

- **C\((i,j)\)** is what it costs factory \(i\) to make product \(j\)
- **Inv\((i,j)\)** is the inventory in factory \(i\) of product \(j\)
- **PO\((j)\)** is the number of product \(j\)'s that the client wants

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>36</td>
<td>22</td>
<td>15</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>35</td>
<td>20</td>
<td>12</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>37</td>
<td>21</td>
<td>16</td>
<td>59</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>5</td>
<td>99</td>
<td>34</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>19</td>
<td>83</td>
<td>12</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>29</td>
<td>21</td>
<td>56</td>
<td>87</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>12</td>
<td>29</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Cost for factory 1:

\[1 \times 10 + 0 \times 36 + 12 \times 22 + 29 \times 15 + 5 \times 62\]
Cost for factory 1:

\[
\begin{array}{c|c|c|c|c}
& 10 & 36 & 22 & 15 \\
\hline
10 & & & & \\
\hline
12 & 35 & 20 & 12 & 66 \\
\hline
13 & 37 & 21 & 16 & 59 \\
\hline
1 & 0 & 12 & 29 & 5 \\
\end{array}
\]

\[
s = 0; \quad \text{Sum of cost}
\]

\[
\text{for } j=1:5
\]

\[
s = s + C(1,j) \times PO(j)
\]

\[
\text{end}
\]
Cost for factory 2:

\[ s = 0; \quad \% \text{Sum of cost} \]
\[ \text{for } j=1:5 \]
\[ s = s + C(2,j) \times \text{PO}(j) \]
\[ \text{end} \]
Cost for factory $i$:

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

\[
iBest = 0; \quad \text{minBill} = \text{inf};
\]

\[
\text{for } i = 1:n\text{Fact}
\]

\[
iBill = \text{iCost}(i,C,PO);
\]

\[
\text{if } iBill < \text{minBill}
\]

\[
% \text{Found an Improvement}
\]

\[
iBest = i; \quad \text{minBill} = \text{iBill};
\]

\[
\text{end}
\]

\[
\text{end}
\]
inf – a special value that can be regarded as positive infinity

\[ x = \frac{10}{0} \text{ assigns inf to } x \]
\[ y = 1 + x \text{ assigns inf to } y \]
\[ z = \frac{1}{x} \text{ assigns 0 to } z \]
\[ w < \text{inf} \text{ is always true if } w \text{ is numeric} \]
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?

<table>
<thead>
<tr>
<th>Inv</th>
<th>38</th>
<th>5</th>
<th>99</th>
<th>34</th>
<th>42</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>82</td>
<td>19</td>
<td>83</td>
<td>12</td>
<td>42</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>29</td>
<td>21</td>
<td>56</td>
<td>87</td>
<td>Yes</td>
</tr>
<tr>
<td>PO</td>
<td>1</td>
<td>0</td>
<td>12</td>
<td>29</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

The table shows who can fill the order based on inventory (Inv) and purchase orders (PO).
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

<table>
<thead>
<tr>
<th>Inv</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>5</td>
<td>99</td>
<td>34</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>19</td>
<td>83</td>
<td>12</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>29</td>
<td>21</td>
<td>56</td>
<td>87</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PO</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>12</td>
<td>29</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Method 1: check the inventory for every product
Initialization

<table>
<thead>
<tr>
<th>Inv</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>38</td>
<td>5</td>
<td>99</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>82</td>
<td>19</td>
<td>83</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>29</td>
<td>21</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>42</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PO</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>12</td>
<td>29</td>
<td>5</td>
</tr>
</tbody>
</table>

DO 1
Still True…

\[
\begin{array}{cccccc}
\text{Inv} & 38 & 5 & 99 & 34 & 42 \\
 & 82 & 19 & 83 & 12 & 42 \\
 & 51 & 29 & 21 & 56 & 87 \\
\end{array}
\]

\[
\begin{array}{cccccc}
\text{PO} & 1 & 0 & 12 & 29 & 5 \\
\end{array}
\]

\[
\text{DO} = \text{DO} \land \land ( \text{Inv}(2,1) \geq \text{PO}(1) )
\]
Still True...

\[
\text{DO} = \text{DO} \land (\text{Inv}(2,2) \geq \text{PO}(2))
\]
Still True…

\[
\begin{array}{cccccc}
\text{Inv} & 38 & 5 & 99 & 34 & 42 \\
 & 82 & 19 & 83 & 12 & 42 \\
 & 51 & 29 & 21 & 56 & 87 \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{PO} & 1 & 0 & 12 & 29 & 5 \\
\end{array}
\]

\[\text{DO} = \text{DO} \land (\text{Inv}(2,3) \geq \text{PO}(3))\]
No Longer True...

\[
\text{DO} = \text{DO} && (\text{Inv}(2,4) \geq \text{PO}(4))
\]
Stay False…

\[
\begin{array}{cccccc}
\text{Inv} & 38 & 5 & 99 & 34 & 42 \\
& 82 & 19 & 83 & 12 & 42 \\
& 51 & 29 & 21 & 56 & 87 \\
\text{PO} & 1 & 0 & 12 & 29 & 5 \\
\end{array}
\]

\[
\text{DO} = \text{DO} && (\text{Inv}(2,5) \geq \text{PO}(5))
\]
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
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…
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);