Previous Lecture:
- 2-d array—matrix

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

Announcement:
- Prelim 1 tonight at 7:30pm
  - Last names A-L in Uris Hall Room G01
  - Last names M-Z in Baker Lab Room 100

Storing and using data in **tables**

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

```
  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?

```
A = (0 0 1 0 1 0)
   (1 0 0 1 1 1)
   (0 1 0 1 1 1)
   (1 0 1 0 1 0)
   (0 0 1 0 1 1)
   (0 1 0 1 0 0)
```

Connections between webpages:

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

Represent the web pages graphically...

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

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

<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>

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

Inventory (or Capacity) Array

<table>
<thead>
<tr>
<th></th>
<th>38</th>
<th>5</th>
<th>99</th>
<th>34</th>
<th>42</th>
</tr>
</thead>
<tbody>
<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>

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

Purchase Order

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

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

Cost for factory $i$:

$$s = 0; \text{ %Sum of cost for } j=1:5$$

$$s = s + C(i,j)*PO(j)$$

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

Encapsulate...
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

- \( x = \frac{10}{0} \) assigns inf to \( x \)
- \( y = 1 + x \) assigns inf to \( y \)
- \( z = \frac{1}{x} \) assigns 0 to \( z \)
- \( w < \text{inf} \) 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?

<table>
<thead>
<tr>
<th>Inv</th>
<th>PO</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>1</td>
</tr>
<tr>
<td>82</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>0</td>
</tr>
<tr>
<td>59</td>
<td>12</td>
</tr>
<tr>
<td>29</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>29</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
</tr>
</tbody>
</table>

Yes

No

Example: Check inventory of factory 2

<table>
<thead>
<tr>
<th>Inv</th>
<th>PO</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>1</td>
</tr>
<tr>
<td>82</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>0</td>
</tr>
<tr>
<td>59</td>
<td>12</td>
</tr>
<tr>
<td>29</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>29</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
</tr>
</tbody>
</table>

Method 1: check the inventory for every product

Wanted: A True/False Function

(\text{DO} \text{ is “true” if factory } i \text{ can fill the order.})

(\text{DO} \text{ is “false” if factory } i \text{ cannot fill the order.})
Still True...

<table>
<thead>
<tr>
<th>Inv</th>
<th>DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>82</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td></td>
</tr>
</tbody>
</table>

PO

| 1 | 0 | 12 | 29 | 5 |

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

Encapsulate...

function \( DO = \text{iCanDo}(i,\text{Inv},\text{PO}) \)

% DO is true if factory \( i \) can fill
% the purchase order. Otherwise, false

\[
n\text{Prod} = \text{length}(\text{PO});
DO = 1;
\text{for } j = 1:n\text{Prod}
    DO = DO \land (\text{Inv}(i,j) \geq \text{PO}(j));
\end{function}

Encapsulate...

function \( DO = \text{iCanDo}(i,\text{Inv},\text{PO}) \)

% DO is true if factory \( i \) can fill
% the purchase order. Otherwise, false

\[
n\text{Prod} = \text{length}(\text{PO});
DO = 1;
\text{for } j = 1:n\text{Prod}
    DO = DO \land (\text{Inv}(i,j) \geq \text{PO}(j));
\end{function}

Back To Finding the Cheapest

\[
i\text{Best} = 0; \text{minBill} = \text{inf};
\text{for } i=1:n\text{Fact}
    \text{if } \text{iCanDo}(i,\text{Inv},\text{PO})
        \text{iBill} = \text{iCost}(i,\text{C},\text{PO});
        \text{if } \text{iBill} < \text{minBill}
            \text{iBest} = i; \text{minBill} = \text{iBill};
    \end{function}
\]

Finding the Cheapest

<table>
<thead>
<tr>
<th>C</th>
<th>10</th>
<th>36</th>
<th>22</th>
<th>15</th>
<th>62</th>
<th>1019</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>35</td>
<td>20</td>
<td>12</td>
<td>66</td>
<td>930</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>37</td>
<td>21</td>
<td>16</td>
<td>59</td>
<td>1040</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{PO} =
| 1 | 0 | 12 | 29 | 5 |
\]

As computed by \( \text{iCost} \)
As computed by \( \text{iCanDo} \)