CSCI-UA.0380-001
Programming Challenges

Sean McIntyre
Class 08: Dynamic Programming II
Dynamic Programming II
Fibonacci sequence

- Y'know, the famous one
  - 1, 1, 2, 3, 5, 8, 13, 21, …
Fibonacci sequence

- As a function:
  - $F(1) = 1$
  - $F(2) = 1$
  - $F(i) = F(i-1) + F(i-2)$ for $i = 3, 4, \ldots$
Fibonacci sequence

- First, the naïve way to compute $F(8)$

$F(8) = ?$
Fibonacci sequence

- First, the naïve way to compute $F(8)$

$F(8) = ?$

$F(7) = ?$

$F(6) = ?$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

• First, the naïve way to compute $F(8)$

```
F(8) = ?
F(7) = ?
F(6) = ?
F(5) = ?
F(4) = ?
```
Fibonacci sequence

- First, the naïve way to compute $F(8)$

```
F(8)=?
F(7)=?
F(6)=?
F(5)=?
F(4)=?
F(3)=?
F(2)=?
F(1)=?
F(0)=?
```
Fibonacci sequence

First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

• First, the naïve way to compute $F(8)$
Fibonacci sequence

• First, the naïve way to compute $F(8)$
Fibonacci sequence

• First, the naïve way to compute $F(8)$

$F(8) = ?$

$F(7) = ?$

$F(6) = ?$

$F(8) = ?$

$F(7) = ?$

$F(7) = ?$

$F(5) = ?$

$F(6) = ?$

$F(7) = ?$

$F(4) = ?$

$F(7) = ?$

$F(2) = 1$

$F(3) = 2$

$F(4) = 3$

$F(5) = ?$

$F(4) = ?$

$F(3) = ?$

$F(7) = ?$

$F(2) = 1$

$F(3) = 2$

$F(2) = 1$
Fibonacci sequence

• First, the naïve way to compute $F(8)$

\[
F(8) = ? \\
F(7) = ? \\
F(6) = ? \\
F(5) = ? \\
F(4) = ? \\
F(3) = ? \\
F(2) = ? \\
F(1) = ? \\
F(0) = 0 \\
F(1) = 1
\]
Fibonacci sequence

- First, the naïve way to compute F(8)
Fibonacci sequence

• First, the naïve way to compute $F(8)$

- $F(1) = 1$
- $F(2) = 1$
- $F(3) = 2$
- $F(4) = 3$
- $F(5) = 5$
- $F(6) = 8$
- $F(7) = 13$
- $F(8) = 21$
Fibonacci sequence

- First, the naïve way to compute $F(8)$

\[
\begin{align*}
F(8) &= \? \\
F(7) &= \? \\
F(6) &= \? \\
F(5) &= \? \\
F(4) &= \? \\
F(3) &= 2 \\
F(2) &= 3
\end{align*}
\]
Fibonacci sequence

First, the naïve way to compute $F(8)$
Fibonacci sequence

• First, the naïve way to compute $F(8)$

F(5)=5

F(4)=?

F(5)=?

F(6)=?

F(7)=?

F(8)=?
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$

```
F(8) = ?
F(7) = ?
F(6) = ?
F(5) = ?
F(4) = ?
F(3) = ?
F(2) = 1
F(1) = 1
F(5) = 5
F(6) = ?
F(7) = ?
F(8) = ?
F(5) = ?
F(4) = ?
F(6) = ?
F(7) = ?
F(8) = ?
```
Fibonacci sequence

• First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute F(8)

\[
\begin{align*}
F(8) &= \, ? \\
F(7) &= \, ? \\
F(6) &= \, ? \\
F(7) &= \, ? \\
F(5) &= \, ? \\
F(6) &= \, ? \\
F(7) &= \, ? \\
F(4) &= \, 3 \\
F(5) &= \, 5 \\
F(3) &= \, 2 \\
F(2) &= \, 1 \\
\end{align*}
\]
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$

```
F(8) = ?
F(7) = ?
F(6) = ?
F(5) = ?
```

F(6) = 8
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute F(8)

F(8) = ?
F(7) = ?
F(6) = 8
F(5) = 5
F(4) = 3
F(3) = 2
F(2) = 2
F(1) = 1
Fibonacci sequence

- First, the naïve way to compute $F(8)$

```
F(8) = ?
F(7) = ?
F(6) = ?
F(5) = 5
F(6) = 8
```
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$
Fibonacci sequence

- First, the naïve way to compute $F(8)$

```
$F(7)$=13
$F(6)$=
$F(5)=?$
$F(4)=?$
```
Fibonacci sequence

- First, the naïve way to compute F(8)

You get the picture...
Fibonacci sequence

- Second, the DP memoized version for F(8)

\[ F(8) = ? \]
Fibonacci sequence

- Second, the DP memoized version for F(8)

F(8)=?
F(7)=?
F(6)=?
Fibonacci sequence

- Second, the DP memoized version for $F(8)$
Fibonacci sequence

- Second, the DP memoized version for F(8)
Fibonacci sequence

- Second, the DP memoized version for $F(8)$
Fibonacci sequence

- Second, the DP memoized version for $F(8)$
Fibonacci sequence

- Second, the DP memoized version for $F(8)$
Fibonacci sequence

- Second, the DP memoized version for F(8)

F(2) = 1
F(1) = 1
F(3) = 2
F(4) = ?
F(5) = ?
F(6) = ?
F(7) = ?
F(8) = ?
Fibonacci sequence

• Second, the DP memoized version for F(8)

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
</tbody>
</table>
**Fibonacci sequence**

- Second, the DP memoized version for $F(8)$

```
<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
</tbody>
</table>
```

![Diagram showing the Fibonacci sequence and memo table](image-url)
Fibonacci sequence

- Second, the DP memoized version for F(8)

```
F(8) = ?
F(7) = ?
F(6) = ?
F(5) = ?
F(4) = ?
F(3) = ?
F(2) = 1
F(1) = 1
```

Memo table:

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Diagram:

- F(3) = 2
- F(2) = 1
- F(4) = 3
- F(5) = ?
- F(6) = ?
- F(7) = ?
- F(8) = ?
Fibonacci sequence

- Second, the DP memoized version for $F(8)$

 Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>$F(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for $F(8)$

<table>
<thead>
<tr>
<th>x</th>
<th>$F(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$F(3)=2$</td>
</tr>
<tr>
<td>4</td>
<td>$F(4)=3$</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for F(8)
Fibonacci sequence

- Second, the DP memoized version for $F(8)$

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for F(8)

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for $F(8)$
Fibonacci sequence

- Second, the DP memoized version for F(8)

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for F(8)

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for $F(8)$

![Fibonacci diagram]

<table>
<thead>
<tr>
<th>Memo table</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
Fibonacci sequence

Second, the DP memoized version for $F(8)$

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
</tbody>
</table>
Fibonacci sequence

• Second, the DP memoized version for F(8)

F(8)=?  F(7)=?  F(6)=?

F(7)=?  F(6)=?  F(5)=?

F(5)=5  F(4)=3

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for $F(8)$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$F(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$F(3)=2$</td>
</tr>
<tr>
<td>4</td>
<td>$F(4)=3$</td>
</tr>
<tr>
<td>5</td>
<td>$F(5)=5$</td>
</tr>
<tr>
<td>6</td>
<td>$F(6)=8$</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for F(8)

```latex
\begin{tabular}{|c|c|}
  \hline
  x & F(x) \\
  \hline
  3 & F(3)=2 \\
  4 & F(4)=3 \\
  5 & F(5)=5 \\
  6 & F(6)=8 \\
  \hline
\end{tabular}
```
Fibonacci sequence

- Second, the DP memoized version for F(8)

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
<tr>
<td>6</td>
<td>F(6)=8</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for $F(8)$

Memo table

<table>
<thead>
<tr>
<th>$x$</th>
<th>$F(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
<tr>
<td>6</td>
<td>F(6)=8</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for $F(8)$

$$
\begin{array}{|c|c|}
\hline
x & F(x) \\
\hline
3 & F(3) = 2 \\
4 & F(4) = 3 \\
5 & F(5) = 5 \\
6 & F(6) = 8 \\
\hline
\end{array}
$$
Fibonacci sequence

- Second, the DP memoized version for F(8)

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
<tr>
<td>6</td>
<td>F(6)=8</td>
</tr>
<tr>
<td>7</td>
<td>F(7)=13</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for $F(8)$

**Memo table**

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$F(3)=2$</td>
</tr>
<tr>
<td>4</td>
<td>$F(4)=3$</td>
</tr>
<tr>
<td>5</td>
<td>$F(5)=5$</td>
</tr>
<tr>
<td>6</td>
<td>$F(6)=8$</td>
</tr>
<tr>
<td>7</td>
<td>$F(7)=13$</td>
</tr>
</tbody>
</table>
Fibonacci sequence

• Second, the DP memoized version for F(8)

F(7) = 13
F(6) = ?
F(8) = ?

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
<tr>
<td>6</td>
<td>F(6)=8</td>
</tr>
<tr>
<td>7</td>
<td>F(7)=13</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for F(8)

**Memo table**

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
<tr>
<td>6</td>
<td>F(6)=8</td>
</tr>
<tr>
<td>7</td>
<td>F(7)=13</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for F(8)

Memo table

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
<tr>
<td>6</td>
<td>F(6)=8</td>
</tr>
<tr>
<td>7</td>
<td>F(7)=13</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- Second, the DP memoized version for F(8)

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>F(3)=2</td>
</tr>
<tr>
<td>4</td>
<td>F(4)=3</td>
</tr>
<tr>
<td>5</td>
<td>F(5)=5</td>
</tr>
<tr>
<td>6</td>
<td>F(6)=8</td>
</tr>
<tr>
<td>7</td>
<td>F(7)=13</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- These methods of computing the Fibonacci sequence use **recursion**
  - Referred to as **top-down**
  - “Find the answer for my problem”
- Another way of computing the Fibonacci sequence is by **iteratively building** the solution
  - Referred to as **bottom-up**
  - “Find the answers for all the sub problems, then find the answer for my problem”
Fibonacci sequence

- As a function:
  - $F(1) = 1$
  - $F(2) = 1$
  - $F(i) = F(i-1) + F(i-2)$ for $i = 3, 4, \ldots$
Fibonacci sequence

• As a table:
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1
Fibonacci sequence

• As a table:
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fibonacci sequence

- As a table:
  - F[1] = 1
  - F[2] = 1
  
<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

- As a table:
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

- As a table:
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

- As a table:
  - `F = new int[9];`
  - `F[1] = 1`
  - `F[2] = 1`

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

• As a table:
  
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fibonacci sequence

- As a table:
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

- As a table:
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

- As a table:
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

- As a table:
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

- As a table:
  - \( F = \text{new int}[9]; \)
  - \( F[1] = 1 \)
  - \( F[2] = 1 \)

<table>
<thead>
<tr>
<th>(x)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F[x])</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

- As a table:
  - $F = \text{new int}[9]$;
  - $F[1] = 1$
  - $F[2] = 1$

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F[x]$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

Fibonacci sequence

- As a table:
  - `F = new int[9];`
  - `F[1] = 1`
  - `F[2] = 1`

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>21</td>
</tr>
</tbody>
</table>

Fibonacci sequence

• As a table:
  – F = new int[9];
  – F[1] = 1
  – F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>21</td>
</tr>
</tbody>
</table>
Fibonacci sequence

- As a table:
  - F = new int[9];
  - F[1] = 1
  - F[2] = 1

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>F[x]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>21</td>
</tr>
</tbody>
</table>

Here's F(8)!
Longest increasing subsequence

- Given an array of numbers,
  - int a = new int[n];
- find the longest increasing subsequence
  - a subset s such that a[s[0]] < a[s[1]] < a[s[2]] < ...
  - And |s| is maximal

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>
Here's an increasing subsequence – Is it the longest increasing subsequence?

To check, we can use bottom-up DP

- Our algorithm will build the solution to all sub problems before finding the solution to the overall problem
- Find the solution for a[0:1], then a[0:2], …, then a[0:n]
Longest increasing subsequence

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

• Algorithm:
  
  − Build a table $s$ so that $s[i]$ is the longest increasing subsequence ending with $a[i]$
  
  • Includes $a[i]$ in the sequence
  
  − $s[i] = \max \left\{ s[j] +1 \text{ if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}$

Builds on prior subsequences
Subsequence of itself
Longest increasing subsequence

\[
s[i] = \max \left\{ s[j] + 1 \text{ if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

- Demo of the algorithm
## Longest increasing subsequence

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>s[i]</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
s[i] = \max \begin{cases} 
  s[j] + 1 & \text{if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \\
  1 & 
\end{cases}
\]

- Demo of the algorithm
Longest increasing subsequence

\[
s[i] = \max\left\{ s[j] + 1 \text{ if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

• Demo of the algorithm
Longest increasing subsequence

\[
s[i] = \max \left\{ s[j] +1 \mid \text{if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

- Demo of the algorithm
### Longest increasing subsequence

The problem of finding the longest increasing subsequence in a given sequence of numbers can be solved using dynamic programming. We define a table \( s[i] \) where \( s[i] \) represents the length of the longest increasing subsequence ending with \( a[i] \). The recurrence relation for \( s[i] \) is given by:

\[
 s[i] = \max \left\{ s[j] + 1 \mid a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

Here's a demonstration of the algorithm with a given sequence:

<table>
<thead>
<tr>
<th>( i )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a[i] )</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>( s[i] )</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Demo of the algorithm
Longest increasing subsequence

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>s[i]</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
s[i] = \max \left\{ n \left| s[j] + 1 \text{ if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right. \right\}
\]

- Demo of the algorithm
Longest increasing subsequence

\[
s[i] = \max \left\{ s[j] + 1 \mid a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

- Demo of the algorithm
Longest increasing subsequence

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>s[i]</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
s[i] = \max \left\{ s[j] + 1 \text{ if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

- Demo of the algorithm
Longest increasing subsequence

\[
s[i] = \max \left\{ s[j] + 1 \mid a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

- Demo of the algorithm
Longest increasing subsequence

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>s[i]</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

\[ s[i] = \max \left\{ s[j] + 1 \text{ if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\} \]

- Demo of the algorithm
# Longest increasing subsequence

\[
s[i] = \max \left\{ s[j] + 1 \mid a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

- Demo of the algorithm

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>s[i]</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
Longest increasing subsequence

\[
s[i] = \max \left\{ s[j] + 1 \text{ if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

- Demo of the algorithm
Longest increasing subsequence

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>s[i]</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>p[i]</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
s[i] = \max \left\{ s[j] + 1 \text{ if } a[i] > a[j] \text{ for } j = 1, 2, \ldots, i-1 \right\}
\]

\[
p[i] = j, \text{ or } -1 \text{ if no } j \text{ exists}
\]

- Demo of the algorithm
  - Keeping track of the longest increasing subsequence
void lisLength(int a[]) {
    int N = a.length;
    int LIS[] = new int[N];

    for (int i = 0; i < N; i++) {
        LIS[i] = 1; // at least 1
        for (int j = 0; j < i; j++) {
            if (a[i] > a[j]) {
                LIS[i] = max(LIS[i], LIS[j]+1); // update with LIS[j]
            }
        }
    }

    int LISlength = 0;
    for (int I = 0; I < N; I++) LISlength = max(LISlength, LIS[I]);
    return LISlength;
}
Is Bigger Smarter?

- UVa 10131
  - Read it, then try to solve it! Hint in 5 minutes.
- Hint: Sort the elephants by weights increasing, if there's a tie then by IQ decreasing
  - This reduces the problem down to longest increasing subsequence
    - Discuss why?
Is Bigger Smarter?

Sort by weight
Then by IQ

Longest decreasing subsequence

Answer!

6008 1300
6000 2100
500 2000
1000 4000
1100 3000
2000 1900
6000 2000
8000 1400
6000 1200
6000 1200
6008 1300
2000 1900
8000 1400
Wedding Shopping

• You are going shopping for a wedding
  – There are 1 ≤ C ≤ 20 types of garments
  – Each garment type has 1 ≤ K ≤ 20 items
    • e.g., 2 types of shirts, 3 different belts, etc.
  – You have a budget of 1 ≤ M ≤ 200
  – Task: Buy one of each type of garment, spending as much money as possible without going over budget
  – What is the maximum possible amount to spend?
## Wedding Shopping

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shirt</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Shoes</td>
<td>50</td>
<td>14</td>
<td>23</td>
<td>8</td>
</tr>
</tbody>
</table>
## Wedding Shopping

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shirt</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Shoes</td>
<td>50</td>
<td>14</td>
<td>23</td>
<td>8</td>
</tr>
</tbody>
</table>

M = 100

Answer: 75
## Wedding Shopping

### Item Costs

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M = 100</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shirt</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Shoes</td>
<td>50</td>
<td>14</td>
<td>23</td>
<td>8</td>
</tr>
</tbody>
</table>

**Answer:** 75

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M = 20</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shirt</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
# Wedding Shopping

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shirt</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Shoes</td>
<td>50</td>
<td>14</td>
<td>23</td>
<td>8</td>
</tr>
</tbody>
</table>

**M = 100**

**Answer:** 75

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shirt</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

**M = 20**

**Answer:** 19

(Multiple solutions)
Wedding Shopping

<table>
<thead>
<tr>
<th>M = 5</th>
<th></th>
<th></th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shirt</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
# Wedding Shopping

<table>
<thead>
<tr>
<th>M = 5</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shirt</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Answer: No solution!
Wedding Shopping

- How did we solve this problem last time?
  - Dynamic programming (top-down)
  - State: (money remaining, garment index)
  - Goal: **Find all reachable states**
    - i.e., find out how much money we have remaining when we have selected some number of garments
    - The answer will be the least amount of money remaining when we have selected all garments
Wedding Shopping

• Bottom-up DP formulation
  
  - Define a table \( dp[moneyRem][i] \) where \( i \) is the garment index
  
  - \( dp[moneyRem][i] \) is true if it's possible to end up with \( moneyRem \) by choosing garments 1 through \( i \)
### Wedding Shopping

Example:

<table>
<thead>
<tr>
<th>M = 12</th>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shirt (1)</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pants (2)</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt (3)</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

dp[ moneyRem ][ i ] is true if it's possible to end up with moneyRem by choosing garments 1 through i

<table>
<thead>
<tr>
<th>dp:</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>moneyRem</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i (garment index)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Wedding Shopping

**Example:**

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M = 12</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shirt (1)</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Pants (2)</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt (3)</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

*dp[ moneyRem ][ i ]* is true if it's possible to end up with *moneyRem* by choosing garments 1 through *i*

**BASE CASE!**
## Wedding Shopping

### Example:

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>M = 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shirt (1)</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Pants (2)</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt (3)</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

dp[ moneyRem ][ i ] is true if it's possible to end up with moneyRem by choosing garments 1 through i

### Table:

<table>
<thead>
<tr>
<th>i (garment index)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>moneyRem</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Wedding Shopping

Example:

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>M = 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shirt (1)</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Pants (2)</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt (3)</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

$\text{dp}[ \text{moneyRem} ][ i ]$ is true if it's possible to end up with moneyRem by choosing garments 1 through $i$.

<table>
<thead>
<tr>
<th>moneyRem</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>T</td>
</tr>
</tbody>
</table>
Wedding Shopping

Example:

<table>
<thead>
<tr>
<th>Item</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>M = 12</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Shirt (1)</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Pants (2)</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belt (3)</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

dp[ moneyRem ][ i ] is true if it's possible to end up with moneyRem by choosing garments 1 through i
Wedding Shopping

- Bottom-up DP formulation
  - Define a table $dp[\text{moneyRem}][i]$ where $i$ is the garment index
  - $dp[\text{moneyRem}][i]$ is true if it's possible to end up with $\text{moneyRem}$ by choosing garments 1 through $i$
  - $dp[\text{moneyRem}][i]$
    \[= \text{OR} \left( dp[\text{moneyRem} - \text{price}_j][i-1] \right) \]
    \[\text{for } j = 0, 1, \ldots, |\text{items}_i| - 1 \]
  - $\text{items}_i$ refers to all the items for the $i$th garment
Wedding Shopping

- This works because we're simply marking all possible states that are reachable.

- The answer to “is it possible to have $x$ remaining on $m$ garments?” is yes if and only if $dp[x][m]$ is true.

- The answer to “what is the most one can spend on all $n$ garments without going over budget?” is:
  - What is the least we can spend on $n$ garments?
  - $\min$(moneyRem for all $dp[moneyRem][n]$ that are true)
Runtime analysis:

- It only takes as long as it takes to fill up each of the states in the dp table.
- \((\text{Amount of money}) \times (\text{number of garments}) = 200 \times 20 = 4000\) state space.
- \((\text{Number of items}) = 20\) operations to fill each state.
- \(4000 \times 20 = 80,000\) operations, small!
• UVa 10192
  – Solve it! Hint after 5 minutes
• “Longest common subsequence”
  – You are given two strings, $S_1$ and $S_2$
  – $dp[i][j]$ is the length of the longest common subsequence after $i$ characters of $S_1$ and $j$ characters of $S_2$
  – $dp[i][j] = \max(\begin{array}{l}
  dp[i-1][j], \\
  dp[i][j-1], \\
  dp[i-1][j-1] + 1
\end{array})$
  \text{ if } S_1[i] = S_2[j]$
```
• Example:
  – abcd and acdb

<table>
<thead>
<tr>
<th></th>
<th>_</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
```

dp[i][j] = max(
  dp[i−1][j],
  dp[i][j−1],
  dp[i−1][j−1] + 1 if S1[i] = S2[j]
)
Vacation

- Why this works (sketch):
  - $dp[i][j]$ contains the largest common subsequence between $S_1[0:i]$ and $S_2[0:j]$
  - At each step, "consume" a character from either string, incrementally build upon the best answer
    - If possible (and if the answer is better), "consume" a character from both and increment the subsolution
  - Therefore the largest common subsequence is in $dp[n][m]$ where $|S_1| = n$ and $|S_2| = m$
Let Me Count The Ways

• Given the coin system \{ 1, 5, 10, 25, 50 \}, how many different combinations of coins can make \( n \) cents?
Let Me Count The Ways

- Sketch of how to do this:
  - Create a 2D int array so that dp[ amount ][ i ] contains the number of ways to make amount using coins 0 through i
  - coins = [ 1, 5, 10, 25, 50 ] (an array)
  - This ensures we are counting combinations and not permutations

\[
\text{dp[ amount ][ i ]} = \begin{cases} 
\text{dp[ amount ][ i-1 ]} + \text{dp[ amount } - \text{coins[ i ]][ i ]} 
\end{cases}
\]
Let Me Count The Ways

<table>
<thead>
<tr>
<th></th>
<th>50</th>
<th>25</th>
<th>10</th>
<th>5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Base case: $dp[ amount ][ i ] = 0$

Result will be here: $dp[ amount ][ 13 ]$

$$dp[ amount ][ i ] = dp[ amount ][ i-1 ] + dp[ amount - coins[ i ] ][ i ]$$
Let Me Count The Ways

Pick zero coins

\[
\text{dp}\left[\text{amount}\right][i] = \begin{cases} 1 & \text{if } i = 0 \\ \text{dp}\left[\text{amount}\right][i-1] + \text{dp}\left[\text{amount} - \text{coins}[i]\right][i] & \text{otherwise} \end{cases}
\]
Let Me Count The Ways

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Pick zero coins

Pick another coin

\[
\text{dp[ amount ][ } i \text{ ]} = \text{dp[ amount ][ } i-1 \text{ ]} + \text{dp[ amount } - \text{coins}[ i ]\text{ ][ } i \text{ ]}
\]
Let Me Count The Ways

\[
\text{dp[ amount ][ i ]} = \ \text{dp[ amount ][ i-1 ]} + \ \text{dp[ amount – coins[ i ]][ i ]}
\]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Let Me Count The Ways

\[
dp[\text{amount}][i] = \begin{cases} 
1 & \text{if } i = 0 \\
\dp[\text{amount}][i-1] + \dp[\text{amount} - \text{coins}[i]][i] & \text{otherwise}
\end{cases}
\]
Let Me Count The Ways

- Runtime analysis:
  - 5 types of coins, given values are at most 30,000 (as per problem statement)
  - $O(1)$ to compute each cell
  - $5 \times 30,000 = 150,000$ operations, cool!
Dynamic Programming

- **Top-down DP**
  - Pro: Natural transformation from recursion
  - Pro: Computes subproblems only when necessary
  - Con: May be slower due to recursion overhead
  - Con: Uses exactly O(states) table size

- **Bottom-up DP**
  - Pro: Faster if many subproblems visited, no recursion
  - Pro: Can save memory space
  - Con: May not be as intuitive
  - Con: Fills values for all the states, does not skip unreachable states
Practice

• Using the UVa system
For next class

- **Readings:**
  - Sections 3.5
    - Look at the classical DP problems that we didn't go over in class

- **Exercises:**
  - Homework on the website
  - Read over the DP section again to see if it makes more sense