Warm-Up Question 1

Fib\( (n : \text{non-negative integer}) \)

\[
\begin{align*}
\text{if } n \text{ is 0 or 1 then} & \\
\quad \text{return } n \\
\text{else} & \\
\quad \text{return } \text{Fib}(n - 1) + \text{Fib}(n - 2)
\end{align*}
\]

Fib(5) calls?
Warm-Up Question 2

- Given $n$ intervals, 1...$n$,
  - each has start time $s_i$ and finish time $f_i$.
- For each interval, compute a value $p[i]$,
  - $p[i] = j$ means $j$ is the latest $f_j$ such that $f_j \leq s_i$.
  - If no intervals end before $s_i$, then $f[i] = 0$.
- Intervals are already sorted by finish time.

Example:

```
1 2
  3
    4
      5
        6
```
Warm-Up Question 2

Warm-up(int $n$, intervals $[s_1, f_1], [s_2, f_2], \ldots [s_n, f_n]$)
Sort intervals by finish time (if not already)
for each interval $[s_i, f_i]$
    binary search for latest $f_j \leq s_i$
Interval Scheduling Problem Statement

- Which classes should take next quarter?
- The classes all meet once a day,
  - at different times and lengths
  - are worth different amounts of credits.
- Maximize amount of credits earned in quarter
- Without having to skip any classes
Interval Scheduling: Recursive Solution

Key: your friend will take class \(i\) xor won’t

\[ WIS(i) \text{ // opt # of credits, intervals } 1 \ldots i \]

// Base Case:
\[ \text{if } (0 == i) \text{ return 0; } \]

// If my friend doesn’t take class \(i\):
\[ a = \text{value_if_not_taken} = WIS(i-1) \]

// If my friend takes class \(i\):
\[ b = \text{value_if_taken} = V_i + WIS(p[i]) \]

//return something:
\[ \text{return } \max(a, b) \]
WIS(i)

if \( i \) is 0 then
    return 0

// value_if_not_taken = WIS(i - 1)
// value_if_taken = \( v_i + WIS(p[i]) \)
return \( \max(WIS(i - 1), v_i + WIS(p[i])) \)

▷ To solve: call WIS(6) for this input.
Interval Scheduling: Iterative Solution

- WIS[i] needs only info from WIS[0 \ldots i - 1]

- We can write an iterative solution.

```
declare WIS[0 \ldots n]
WIS[0] = 0
for i = 1 to n
WIS[i] = max(WIS[i-1], Vi + WIS[p[i]])
```
### Interval Scheduling: Table

<table>
<thead>
<tr>
<th>$i$</th>
<th>$p[i]$</th>
<th>$WIS(p(i)) + v_i$</th>
<th>$WIS(i - 1)$</th>
<th>$WIS(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0 + 2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0 + 4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2 + 4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0 + 7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>6 + 2</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>6 + 1</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
What classes to take?

- Now we have \( WIS[...] \) filled in.
- Instead of return \( WIS[n] \), output courses.
- Hint: take course \( n \) or no?

\[
i \leftarrow n
\]

\[
\text{while } i > 0:\]
\[
\text{if } WIS[i] == WIS[i-1] : i --
\]
\[
\text{else: output } i
\]
\[
i = p[i]
\]
If asked for a dynamic programming solution:

- Describe *in English* the function
  - Not *how* it works (yet)
  - Yes *what it solves*.
  - Skipping this step = 0 on problem

- Give that function a meaningful variable name.
  - Not “OPT” or “DP” or “table”
  - Not a single letter either.

- Give recursive formulation.