When developing a dynamic-programming algorithm, we follow a sequence of four steps:

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a bottom-up fashion.
4. Construct an optimal solution from computed information.

1. Identify the parameters of the problem
   1. subproblems of the problem

2. Write a dumb recursive algorithm.
   \[ \text{Fib}(n) \]
   \[
   \text{return } \text{Fib}(n-2) + \text{Fib}(n-1)
   \]
   inefficient exponential

3. Make the recursive alg more efficient!
Rod Cutting Choices and Subproblems

Given a rod that is $n$ inches long, what initial choice (or decision) do we need to make?

Should I cut it?

Where to cut first?

1 inch, 2 inch, 3 inch, ..., $n$ inches

Given a choice, what will be the subproblem(s) to solve?

- Decide not to cut — no subproblems
- Make a cut — same as first
- Second piece will be a subproblem
- One piece will no longer be cut

This is the parameter for recursion

$n$ is smaller
Rod Cutting Recursion (Exponential)

Definition
Steps
Steps
Rod Cutting
Rod Cutting 1
▶ Rod Cutting 2
Rod Cutting 3
Rod Cutting 4
Matrix Chain 1
Matrix Chain 2
LCS 1
LCS 2
LCS 3

---

\( p \) is an array of prices
\( n \) is length of the rod

\[
\text{CUT-ROD}_x(p, n)
\]

1. if \( n == 0 \)
2. return 0
3. \( q = \sum_{i=1}^{n} p[i] \)
4. for \( i = 1 \) to \( n \)
5. \( q = \max(q, p[i] + \text{CUT-ROD}_x(p, n - i)) \)
6. return \( q \)

\( q \) is best total value
Rod Cutting Recursion

Definition
Steps
Rod Cutting
Rod Cutting 1
▷ Rod Cutting 2
Rod Cutting 3
Rod Cutting 4
Matrix Chain 1
Matrix Chain 2
LCS 1
LCS 2
LCS 3

Before initial call to CUT-ROD
Create \( T[0... \text{length of rod}] \)
initialize to \(-1\)

\[
\text{CUT-ROD}^M(p, n) \\
\begin{align*}
1 & \quad \text{if } n == 0 \quad \Rightarrow \quad T[n] = 0 \\
2 & \quad \text{return } 0 \\
3 & \quad q = \max \{ p[i] \} \\
4 & \quad \text{for } i = 1 \text{ to } n - 1 \\
5 & \quad q = \max(q, p[i] + \text{CUT-ROD}^M(p, n - i)) \\
6 & \quad \text{return } q
\end{align*}
\]

\( T[n] = ? \)
\( \Theta(n^2) \) I think, need to do \( \Theta(n) \) for loop for \( n \) values
Rod Cutting Recursion

Definition
Steps
Rod Cutting
Rod Cutting 1
  ▶ Rod Cutting 2
Rod Cutting 3
Rod Cutting 4
Matrix Chain 1
Matrix Chain 2
LCS 1
LCS 2
LCS 3

create $T[0...\text{length of rod}]$ array
for $j = 0$ to length of rod
  $T[j] = \text{Cut-Rod}(p, j)$

$\mathcal{B}$

\textbf{CUT-ROD}(p, n)
1  if $n == 0$
2    return 0
3  $q = \prod_{i \leq n} p[i]$
4  for $i = 1$ to $n - 1$
5    $q = \max(q, p[i] + \text{CUT-ROD}(p, n - i))$
6  return $q$

This is a $\Theta(n^2)$ time to do $\Theta(n)$ loop
for $n$ values

CS 3343 Analysis of Algorithms
Dynamic Programming – 7
Exponential Version

\[ \text{Fib}(n) \]
- if \( n = 0 \) or \( n = 1 \)
  - return \( n \)
- else
  - return \( \text{Fib}(n-1) + \text{Fib}(n-2) \)

[Diagram]

The diagram shows the recursive calls for calculating \( \text{Fib}(n) \) for various values of \( n \), with branches representing recursive calls. The diagram highlights the elimination of redundant calculations by memoization.
Memoized version

\[\text{Fib}(n)\]

\[
\begin{align*}
\text{if } T[n] \geq 0 \\
\quad \text{return } T[n] \\
\text{if } n = 0 \text{ or } n = 1 \\
\quad \text{return } T[n] = n \\
\text{return } n \\
\text{else } T[n] \leftarrow \text{Fib}(n-1) + \text{Fib}(n-2) \\
\quad \text{return } T[n] \\
\end{align*}
\]

Bottom-Up Version

\[\text{Fib}(n)\]

\[
\begin{align*}
\text{for } j = 0 \text{ to } n \\
T[2;j] = \text{Fib}[j] \\
\text{if } n = 0 \text{ or } n = 1 \\
\quad \text{return } n \\
\text{else return } T[n-1] + T[n-2] \\
\end{align*}
\]