Divide-and-Conquer (DAC) Limitations

DAC strategy: Subdivide problem P into
\[ P_1, P_2, \ldots, P_k \]
smaller subproblems
solve them recursively, combine solutions

Ex: Fibonacci numbers:
\[ F(n) = F(n-1) + F(n-2) \quad \text{for } n > 1 \]
\[ F(1) = 1, \quad F(0) = 0. \]

Recursive function to calculate \( F \):
\[
\begin{align*}
\text{if } n = 0 \text{ or } n = 1 \text{ then} & \quad \text{return } n \\
\text{else return } f(n-1) + f(n-2) & \\
\end{align*}
\]

Reason why \( RT \) is exponential: same subproblem solved multiple times.

\[ f(n) - \text{non-recursive version: } \quad RT = O(n) \]
\[ F[0] \leftarrow 0, \quad F[1] \leftarrow 1 \]
\[ \text{for } i \leftarrow 2 \text{ to } n \text{ do} \]
\[ F[i] \leftarrow F[i-1] + F[i-2] \]
Rod-Cutting Problem (0-1 Knapsack)

Given a rod of length $n$, and prices $P_1, P_2, \ldots, P_n$ at which each rod of length $i$, $i=1\ldots n$, can be sold, find the best way to cut the given rod to generate maximum total revenue.

Example: $n=10$

<table>
<thead>
<tr>
<th>$i$</th>
<th>1 2 3 4 5 6 7 8 9 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_i$</td>
<td>1 5 8 10 13 17 17 20 24 30</td>
</tr>
</tbody>
</table>

In general, greedy strategy of choosing length with best $\frac{P_i}{i}$ does not yield an optimal solution.

DAC:
Suppose the first piece of an optimal solution is $i$.

Define $v(n) = \text{Max revenue from a rod of length } n$.

$$v(n) = v(n-i) + P_i$$

We don't know the correct $i$ to choose. Try all of them.

$$v(n) = \max_{1 \leq i \leq n} \{ v(n-i) + P_i \} , \quad v(0) = 0$$
Correctness of recurrence: Let $\text{Opt}(n)$ be optimal profit with rod of length $n$ (max profit).

(i) Feasibility: some feasible solution with value $r(n)$

$\text{Opt}(n) \geq r(n)$

Proof by induction on $n$.

Base: $n=0$ $r(0)=0$ $\text{Opt}(0)=0$

Step: $r(n) = r(n-i^*) + p_{i^*}$

$i^* =$ value $i$ that gave max for $r(n)$

$n-i^* < n$ — by Ind: $\text{Opt}(n-i^*) \geq r(n-i^*)$

$\text{Opt}(n) \geq \text{Opt}(n-i^*) + p_{i^*} = r(n)$.

(ii) Optimality: $r(n) \geq \text{Opt}(n)$

Proof by induction on $n$.

Base: $n=0$ $\text{Opt}(0) = r(0) = 0$.

Step: $\text{Opt}(n) =$ first piece has length $i$

$\text{Opt}(n) = \text{Opt}(n-i) + p_i$

$r(n)$ considers $r(n-i) + p_i$ also (tries all $i$)

by Ind: $r(n-i) \geq \text{Opt}(n-i)$

$r(n) \geq r(n-i) + p_i \geq \text{Opt}(n-i) + p_i$

Combining the two, $r(n) = \text{Opt}(n) = \text{Opt}(n)$. 
Implemented recursively, \( R^T = \Omega(c^n) \) for some \( c > 1 \).

Subproblems solved: \( R^0, R^1, \ldots, R^n \)

Store \( R^i \) in \( R[i] \)

Solve problem in increasing values of \( i \):

**DP for rod-cutting problem** \( R^T = O(n^2) \)

\[ R[0] \leftarrow 0 \quad // \text{Base case} \]

\[ \text{for } i \leftarrow 1 \text{ to } n \text{ do} \quad // \text{Rod of length } i \]

\[ x \leftarrow -\infty \]

\[ \text{for } j \leftarrow 1 \text{ to } i \text{ do} \]

\[ x \leftarrow \max \{x, R[i-j] + p_j \} \]

\[ R[i] \leftarrow x \]

Output \( R \)
Alternate Solution for rod-cutting problem

(i) Define \( g(k, i) = \max \) revenue from a rod of length \( k \), cutting pieces of length \( 1, 2, \ldots, i \)

\[
 r(n) = g(n, n)
\]

Recurrence for \( g \):

\[
g(k, i) = \max \left\{ g(k-i, i) + p_i, \quad g(k, i-1) \right\}
\]

\[
g(k, 0) = \begin{cases} 0 & \text{if } k \geq 0 \\ -\infty & \text{if } k < 0 \end{cases}
\]

Redone:

\[
g(k, i) = \begin{cases} g(k, i-1) & \text{if } k < i \\ \max \left\{ g(k-i, i) + p_i, \quad g(k, i-1) \right\} & \text{if } k \geq i \end{cases}
\]

(ii) Another recurrence for \( g \):

Let \( x_i = \# \) of rods of length \( i \) to cut

\[
g(k, i) = \max \left\{ g(k-x_i \cdot i, i-1) + p_i \cdot x_i \right\}
\]

\[
g(k, 0) = 0 \quad k \geq 0.
\]