ACGTCG \text{ } x_1 \ldots n

AATTGC \text{ } y_1 \ldots m

(i) \text{ } x_n = y_m \text{ } \text{LCS}(x, y) = \text{LCS}(x_{1 \ldots n-1}, y_{1 \ldots m-1}) + 1

(ii) \text{ } x_n \neq y_m \text{ } \text{both } x \text{ and } y \text{ are not last char.}
\text{LCS}(x, y) = \max \left( \text{LCS}(x_{1 \ldots n-1}, y) , \text{LCS}(x, y_{1 \ldots m-1}) \right)

\textbf{Recursion for length}

\begin{align*}
c[i,j] &= \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0, \\
\text{c[i-1,j-1]} + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j , \\
\max(c[i,j-1], c[i-1,j]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j .
\end{cases}
\end{align*}

C[i,j] = \text{length of LCS}(x_{1 \ldots i}, y_{1 \ldots j})
Recursion for length

\[ c[i,j] = \begin{cases} 
  0 & \text{if } i = 0 \text{ or } j = 0 , \\
  c[i - 1, j - 1] + 1 & \text{if } i,j > 0 \text{ and } x_i = y_j , \\
  \max(c[i,j - 1], c[i - 1,j]) & \text{if } i,j > 0 \text{ and } x_i \neq y_j .
\end{cases} \]

\( c[i,j] \) is the length of the ICS of \( X_i \text{ and } Y_j \).

Greedy Algorithms

- easy to design
- not always correct
- hard to identify when greedy is the right solution

Greedy alg makes its next step based only on prev. steps (current state) and "simple" calculations over the input.
Rod Cutting is not greedy

\( (A_1, A_2) \quad (A_3, A_4, A_5, A_6) \)

Greedy: Change w/ US coins

Room Scheduling

Alg: Pick smallest interval, recurse
Greedy

Consider a set of requests for a room. Only one person can reserve the room at a time, and you want to allow the maximum number of requests. The requests for periods \((s_i, f_i)\) are:

\[(1, 4), (3, 5), (0, 6), (5, 7), (3, 8), (5, 9), (6, 10), (8, 11), (8, 12), (2, 13), (12, 14)\]

Which ones should we schedule?
Start at end, choose last finishing segment, recurse.

Start at beginning, choose earliest start.

Pick segment w/ min # of overlaps.

Alg: Start at beginning, choose earliest finishing segment; recurse.

Prove a greedy alg is opt., I need
1) optimal substructure
2) greedy choice property

An opt sol'n consistent w/ my greedy choice in the first step.
Opt. Substructure

- Can write down opt. subst. proof
- $m(i,j)$ = # of intervals in an opt. solution from $i$ to $j$

$m(i,j) = \max_{\text{intervals } K \text{ from } i \text{ to } j} \left[ m(i, s_K) + m(f_K, j) + 1 \right]$

Green choice property:

Let $A_m = (s_m, t_m)$ be the activity with the earliest finishing time. Then
1) $\exists$ an optimal solution using $A_m$
2) $(i, s_m)$ is empty.

PF: Suppose we have an opt. solution without $A_m$.
(We need to show how to convert $X$ into another solution that uses $A_m$ and is at least as good)

\[ a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow X \]

Then $X - a_1 + A_m$ is also a valid solution (because $f_m \leq s_j$ $(j \geq 2)$)
and it schedule as many intervals as $X$. $\square$
Greedy

1) Identify opt. substructure
2) Cast problem as a greedy
   choice & prove g.c. property
3) (Write a simple iterative alg.)

---

Robbery

Rob a house, I have a knapsack.
Fill the knapsack with the most profitable items.

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Knapsack Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>B = 50</td>
</tr>
<tr>
<td>Value</td>
<td>60</td>
<td>100</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>Value/Weight</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Integral knapsack - take an item or leave it
Fractional knapsack - take a fraction of an item (infinitely divisible)