Recap: Dynamic Programming (DP)

- DP ≈ "careful brute force"
- DP ≈ subproblems + reuse

time = # subproblems · time/subproblem (treating recursive calls as $\Theta(1)$)

▷ Rod Cutting

Given a n-feet rod, we want to cut it and sell it according to the following prices

length i	1	2	3	4	5	6	7	8	9
price p_i	1	5	8	9	10	17	17	20	24

"Optimal Substructure": optimal solution to a problem utilizes optimal solutions to related subproblems

$$r_n = \max(p_n, r_1 + r_{n-1}, r_2 + r_{n-2}, \ldots, r_{n-1} + r_1)$$

don't cut at all — make cutting after 1 — cutting after 2

Or:

$$r_n = \max(p_1 + r_{n-1}, p_2 + r_{n-2}, \ldots, p_{n-1} + r_1, p_n)$$

Bottom-up version:

```
CutRod(n)
    r[0] = 0
    For i = 1 to n
        q = -∞
        For j = 1 to i
            q = max(q, p_j + r[i-j])
        r[i] = q
    Return r[n]
```

Again, running time is $O(n^2)$.

<u>Note</u>: This outputs the revenue only. What if we want to output how to cut?
Memorize "the cut" for each $r[i]$.

1. Longest Common Subsequence (LCS)

- Subsequence: A is a subsequence of B if
  the characters in A appear in
  order in B

  e.g. subsequences of "ALGORITHM":
  "ALG", "LIT", "AM", "ART" ...

- Common: A is both subsequence of B and C

  e.g. "ALGORITHM" "ARTICHOKE"
  have common subsequences "A", "TH", "ART",
  "RTH", "ARTH", etc.
- Longest: Given two strings, interested in the
  <u>longest</u> common subsequence.
- Applications?
  Similarity between DNA sequences
  Version control (Github)

LCS: Given two strings $x \in \Sigma^m$, $y \in \Sigma^n$,
find their longest common subsequence.
(Σ is the alphabet)

Brute Force?

1. Find all subsequences of x and y respectively.
2. and then find their intersection,
3. and then find the longest one.

Runtime: ① $2^m + 2^n$
② $\Omega(2^m + 2^n)$
③ $\Omega(k)$, k is size of the intersection

Overall runtime is exponential!

Observation:

GUACAMOLE GUANCIALE

1. The LCS must end with "E"
2. The LCS must end with "LE"
3. The LCS for "CAMO" and "NCIA" must be either the LCS for "CAM" and "NCIA" or the LCS for "CAMO" and "NCI".

Formalizing these intuitions:

Thm ("Optimal Substructure"): $X = x_1 x_2 \dots x_m$, $Y = y_1 y_2 \dots y_n$, and $Z = z_1 z_2 \dots z_k$ is an LCS of X and Y.

1. If $x_m = y_n$, then $z_k = x_m = y_n$, and $Z_{[1:k-1]}$ is an LCS of $X_{[1:m-1]}$ & $Y_{[1:n-1]}$
2. If $x_m \neq y_n$, then
   a) If $z_k \neq x_m$, then Z is an LCS of $X_{[1:m-1]}$ and Y.
   b) If $z_k \neq y_n$, then Z is an LCS of X and $Y_{[1:n-1]}$.

Proof: 1. If $z_k \neq x_m$, and Z is an LCS. Consider $Z \| x_m$, this is also a common subsequence of X and Y, but longer by 1. Contradiction! Now we have $z_k = x_m = y_n$, then notice $Z_{[1:k-1]}$ is a common subsequence of $X_{[1:m-1]}$ and $Y_{[1:n-1]}$. We argue why it's the longest. If not, then there's some Z' with $|Z'| > k-1$. $Z' \| x_m$ will be a longer common subsequence than Z, contradiction!

2. a. Since $z_k \neq x_m$, then Z must be a common subsequence of $X_{[1:m-1]}$ and Y. Say it's not the longest, i.e. $\exists Z'$ w/ $|Z'| > |Z|$, then Z' is also a common subsequence of X and Y, but longer! Contradiction!

   b. Similar argument

DP Algorithm: two-dimensional!

1. Subproblems: For $i = 0, 1, \dots, m$ and $j = 0, 1, \dots, n$, let $lcs[i,j]$ be the length of LCS of $x_{[1:i]}$ and $y_{[1:j]}$.

2. Guess: In the case where $x_j \neq y_i$, they can't both be used in the LCS. Which one is not used? we make a guess here.

3. Recurrence:

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

4. Naive Recursion has exponential runtime.
   But we use memoization / bottom-up.
   # of subproblems = (# of i) $\cdot$ (# of j) = $O(m \cdot n)$
   time / subproblem = $O(1)$

   $\Rightarrow$ Runtime is $O(mn)$, w/ memoization or bottom-up.

5.