Longest Common Subsequence
Given two strings, $x$ of length $m$ and $y$ of length $n$, find a common (non-contiguous) subsequence that is as long as possible.

$x = \text{ABCBDAB}$

$y = \text{BDCABA}$

$LCS = \text{BCBA} \text{ or } \text{BCAB}$
Recursive Approach

$LCS(i, j)$ means length of LCS considering only up to $x_i$ and $y_j$

$$LCS(i, j) = \begin{cases} 
0 & \text{if } i \text{ or } j = 0 \\
LCS(i - 1, j - 1) + 1 & \text{if } x_i = y_j \\
\max(LCS(i - 1, j), LCS(i, j - 1)) & \text{otherwise}
\end{cases}$$
Summary of Dynamic Programming

1. optimal substructure: global optimum uses optimal solutions of subproblems
2. ordering over subproblems: solve ‘smallest’ first, build ‘larger’ from them
3. ‘overlapping’ subproblems: polynomial number of subproblems, each possibly used multiple times
4. independent subproblems: optimal solution of one subproblem doesn’t affect optimality of another

- top-down: memoization
- bottom-up: compute table, then recover solution
global optimum uses optimal solutions of subproblems

Proof by contradiction: What if subsolution were not optimal?

Let $z$ be an $LCS(i, j)$ of length $k$.

1. If $x_i = y_j$, then $z_k = x_i = y_j$ and
   
   $LCS(i - 1, j - 1) = z_0...z_{k-1}$.
   
   Not including $z_k$ makes LCS suboptimal: contradiction!

   If $z_0...z_{k-1}$ were not LCS, $z$ could be longer, hence not optimal: contradiction!

2. If $x_i \neq y_j$ and $z_k \neq x_i$, then $z$ is $LCS(i - 1, j)$.
   
   If longer exists, $z$ would not be an LCS: contradiction!

3. If $x_i \neq y_j$ and $z_k \neq y_j$, then $z$ is $LCS(i, j - 1)$
   
   Similar to 2.
- asst 5
- asst 6
- midterm
Knapsack
Given \( n \) objects with integer weights \( w_i \) and values \( v_i \), what is the most valuable subset that weighs at most \( W \)?
Given \( n \) objects with integer weights \( w_i \) and values \( v_i \), what is the most valuable subset that weighs at most \( W \)?

Give an algorithm that runs in \( O(nW) \) time.
Given \( n \) objects with integer weights \( w_i \) and values \( v_i \), what is the most valuable subset that weighs at most \( W \)?

Give an algorithm that runs in \( O(nW) \) time.

Will greedy work? What if items can be divided?
what is the length of the input?
what is the length of the input?

*pseudo-polynomial time*: polynomial if the magnitude of the input numbers is polynomial in the input size.
what is the length of the input? 

*pseudo-polynomial time*: polynomial if the magnitude of the input numbers is polynomial in the input size.

Does this apply to radix sort?
Three-dimensional Dynamic Programming
Given:

1. A context-free grammar $G$ in Chomsky Normal Form
   - $\epsilon$-free
   - all rules like:
     - $A \rightarrow BC$
     - $A \rightarrow a$
   
   So $A \rightarrow BCD$ becomes $A \rightarrow BX$ and $X \rightarrow CD$

2. String of length $n$ of tokens $w_i$
Bottom-up dynamic programming:

\[ \pi[i, j, A] = \text{can } A \text{ span } i-j? \]
\[ \pi[i, i, A] = \]
Bottom-up dynamic programming:

\[ \pi[i, j, A] = \text{can } A \text{ span } i-j? \]
\[ \pi[i, i, A] = \text{true iff } A \rightarrow w_i \]
\[ \pi[i, j, A] = \]
Bottom-up dynamic programming:

\[
\pi[i, j, A] = \text{can } A \text{ span } i-j? \\
\pi[i, i, A] = \text{true iff } A \to w_i \\
\pi[i, j, A] = \bigvee_{A \to BC} \bigvee_{i \leq k < j} \pi[i, k, B] \land \pi[k + 1, j, C]
\]
1. initialize $\pi$ to false everywhere
2. for $i$ from 1 to $n$
3.   foreach nonterminal $A$
4.     $\pi[i, i, A] \leftarrow \text{true if } A \rightarrow w_i$
5. for $\text{len}$ from 2 to $n$
6.   for $i$ from 1 to $n - (\text{len} - 1)$
7.     $j \leftarrow i + (\text{len} - 1)$
8.   for $k$ from $i$ to $j - 1$
9.     foreach rule $A \rightarrow BC$
10.    if $\pi[i, k, B]$ and $\pi[k + 1, j, C]$ then
11.    $\pi[i, j, A] \leftarrow \text{true}$
12. return $\pi[1, n, S]$
For example:

- What’s still confusing?
- What question didn’t you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*