Dynamic Programming - Longest Increasing Subsequence

23, 3, 5, 18, 10, 101, 12, 14

size = 5

Could try including/excluding every element:

$2^n$ subsequences to check
Dynamic Programming - Longest Increasing Subsequence

$S: 23, 3, 5, 18, 10, 101, 12, 14$

$L(S) = 3, 5, 10, 12, 14$ \quad |L(S)| = 5

For dynamic programming we would like

- a recursive expression w/ repeated subproblems
- an easy, fast way to use solved subproblems
\[ L(S) \sim L_{1...n}(S) \sim L_n \]

|Ln| using |Ln-1| ?

\begin{align*}
\text{find } |L_n| \\
\text{if } S[n] > \text{last element in } L_{n-1} \quad \text{then } |L_n| = |L_{n-1}| + 1 \\
\Rightarrow \text{could be at any position in } S
\end{align*}

else \quad \backslash \quad S[n] \leq \text{last}(L_{n-1})

\rightarrow \text{keep } |L_{n-1}| ?

\rightarrow \text{add } S[n] \text{ to suboptimal solution from } S[1...n-1]?
Redefine: \( L_n = \text{longest increasing subsequence that actually uses} \ S[n] \)

\[
L_6 = 1
\]

\[
3, 4, 5, 10, 13, 2
\]

\[
|L_n| = 1 + \max \{ \text{all } j \text{ s.t. } S[j] < S[n] \} |L_j|
\]

look at all \( L_j \) (\( j < n \))
23, 3, 5, 18, 10, 101, 12, 14, 4
1 1 2 3 3 4 4 5 2 → Score may decrease

\[ |L_n| = 1 + \max_{\{\text{all } j \text{ s.t. } S[j] < S[n]\}} |L_j| \]

Recursion: \( L_n \)

BAD

\( L_{n-1} \) \( L_{n-2} \) \( \cdots \) \( L_1 \)

\( L_{n-2} \) \( L_{n-3} \) \( \cdots \) etc

Dyn. Prog: Build solutions, "bottom up"

When it's time to solve \( |L_k| \) we have stored all \( |L_j| \) (\( j < k \)) in an array.

\[ T(k) = \Theta(k) \]

\[ T(n) = \sum_{i=1}^{n} T(k) = \Theta(n^2) \]

Space = \( \Theta(n) \)
$T(n) = \Theta(n^2)$

$\text{space} = \Theta(n)$

$$|L_n| = 1 + \max_{\{\text{all } j \text{ s.t. } S[j] < S[n]\}} |L_j|$$

What about $|L.I.S.|$? $= |L_n|$? $\rightarrow$ NO. $= \max_{j=1..n} |L_j|$

What about $L.I.S.$? Keep the pointers: for each $S[j]$ store any $S[i]$ pointer that generated $|L_j|$
A quick solution for L.I.S. ... but still \( O(n^2) \) & dyn-prog.

\[
\begin{align*}
23, & \quad 3, \quad 5, \quad 18, \quad 10, \quad 101, \quad 12, \quad 14 : S \\
\text{sort} \downarrow & \\
3, & \quad 5, \quad 10, \quad 12, \quad 14, \quad 18, \quad 23, \quad 101 : S_2
\end{align*}
\]

Find longest common subsequence!

- any common subsequence is increasing (assume no duplicates) or remove them
- so LCS\((S,S_2)\) qualifies as a solution
- L1S must exist in \( S_2 \), so it is a candidate for LCS.