Dynamic Programming - Longest Increasing Subsequence

S: 23 3 5 18 10 101 12 14 4 105
Dynamic Programming - Longest Increasing Subsequence

$S: 23 \ 3 \ 5 \ 18 \ 10 \ 101 \ 12 \ 14 \ 4 \ 105$

- Increasing subsequence: $23 \ 101 \ 105$

Size $= 3$

we can do better...
Dynamic Programming – Longest Increasing Subsequence

S: 23 3 5 18 10 101 12 14 4 105
Dynamic Programming – Longest Increasing Subsequence

\[ S : \quad 23 \quad 3 \quad 5 \quad 18 \quad 10 \quad 101 \quad 12 \quad 14 \quad 4 \quad 105 \]
Dynamic Programming - Longest Increasing Subsequence

S: 23 3 5 18 10 101 12 14 4 105
Dynamic Programming - Longest Increasing Subsequence

$S: \ 23 \ 3 \ 5 \ 18 \ 10 \ 101 \ 12 \ 14 \ 4 \ 105$

\[\downarrow\]

$3 \ 5 \ 18 \ 101 \ 105 \ \rightarrow \ \text{size 5}$

Better! But we can still do even better...
Dynamic Programming - Longest Increasing Subsequence

S: 23 3 5 18 10 101 12 14 4 105
Dynamic Programming — Longest Increasing Subsequence

\[ S: \ 23 \ 3 \ 5 \ 18 \ 10 \ 101 \ 12 \ 14 \ 4 \ 105 \]

Longest increasing Subsequence \[ = L(S) = 3 \rightarrow 5 \rightarrow 10 \rightarrow 12 \rightarrow 14 \rightarrow 105 \]

Size = 6
Dynamic Programming – Longest Increasing Subsequence

S: 23 3 5 18 10 101 12 14 4 105

- 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 4 105

Try Dynamic Programming!

Which means we need:

- recursive expression with repeated subproblems
- store solutions to subproblems so that they are quickly accessible
$S: \ 1 \ 23 \ 3 \ 5 \ 18 \ 10 \ 101 \ 12 \ 14 \ 4 \ n^{n-1} \ n$

$\text{Ln}(S):$ longest increasing subsequence that uses $S(n)$
$S: \quad 23 \quad 3 \quad 5 \quad 18 \quad 10 \quad 101 \quad 12 \quad 14 \quad 4 \quad 105$

$\text{Ln}(S):$ longest increasing subsequence that uses $S(n)$

ex. $L_{n-2}(S) = 3, 5, 10, 12, 14 \quad |L_{n-2}| = 5$
$S : \quad 23 \quad 3 \quad 5 \quad 18 \quad 10 \quad 101 \quad 12 \quad 14 \quad 4 \quad 105$

$L_n(S) : \quad \text{longest increasing subsequence that uses} \ \ S(n)$

ex. $L_{n-2}(S) = 3, 5, 10, 12, 14 \quad |L_{n-2}| = 5$

$L_{n-1}(S) = \ ? \ ?$
Let $S = 23, 3, 5, 18, 10, 101, 12, 14, 4, 105$.

$L_n(S)$: longest increasing subsequence that uses $S(n)$.

Example:

$L_{n-2}(S) = 3, 5, 10, 12, 14$  \( |L_{n-2}| = 5 \)

$L_{n-1}(S) = 3, 4$  \( |L_{n-1}| = 2 \)
$S: \begin{array}{cccccccccc}
2 & 3 & 5 & 18 & 10 & 101 & 12 & 14 & 4 & 105
\end{array}$

$L_n(S)$: longest increasing subsequence that uses $S(n)$

ex. $L_{n-2}(S) = 3, 5, 10, 12, 14 \quad |L_{n-2}| = 5$

$L_{n-1}(S) = 3, 4 \quad |L_{n-1}| = 2$

$L_n(S) = ??$
$S: \quad 23 \quad 3 \quad 5 \quad 18 \quad 10 \quad 101 \quad 12 \quad 14 \quad 4 \quad 105$

$\text{Ln}(S) : \quad \text{longest increasing subsequence that uses } S(n)$

ex. $\text{Ln}_{-2}(S) = 3, 5, 10, 12, 14 \quad |\text{Ln}_{-2}| = 5$

$\text{Ln}_{-1}(S) = 3, 4 \quad |\text{Ln}_{-1}| = 2$

$\text{Ln}(S) = 3, 5, 10, 12, 14, 105$
$S: \ 23 \ 3 \ 5 \ 18 \ 10 \ 101 \ 12 \ 14 \ 4 \ 105$

$L_n(S):$ longest increasing subsequence that uses $S(n)$

ex. $L_{n-2}(S) = 3, 5, 10, 12, 14 \quad |L_{n-2}| = 5$

$L_{n-1}(S) = 3, 4 \quad |L_{n-1}| = 2$

$L_n(S) = 3, 5, 10, 12, 14, 105 \quad |L_n| = 1 + |L_{n-2}|$

$L_{n-2}(S)$
$S: \ 1 \ 2 \ 3 \ 5 \ 18 \ 10 \ 101 \ 12 \ 14 \ 4 \ 105$

$L_n(S):$ longest increasing subsequence that uses $S(n)$

$|L_n| = ??$
$S: \quad 23 \quad 3 \quad 5 \quad 18 \quad 10 \quad 101 \quad 12 \quad 14 \quad 4 \quad 105$

$L_n(S):$ longest increasing subsequence that uses $S(n)$

$|L_n| = ??$

look at all $L_j \quad (j < n)$
$S: \begin{array}{ccccccccc} 1 & 2 & 3 & 5 & 18 & 10 & 101 & 12 & 14 & 4 & 105 \end{array}$

$L_n(S)$: longest increasing subsequence that uses $S(n)$

$|L_n| = 1 + ?$

look at all $L_j \ (j < n)$
$S: \ 23 \ 3 \ 5 \ 18 \ 10 \ 101 \ 12 \ 14 \ 4 \ 105$

$L_n(S):$ longest increasing subsequence that uses $S(n)$

$$|L_n| = 1 + \max \{|L_j|\} \{\text{all } j \text{ s.t. } S(j) \leq S(n)\}$$

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

Recursion:

BAD

\[
\begin{align*}
L_n & \quad \quad \quad \quad \quad \quad \\
L_{n-1} & \quad L_{n-2} \quad \ldots \quad L_1 \\
L_{n-2} & \quad L_{n-3} \quad \ldots \\
& \quad \ldots \\
& \quad \text{etc}
\end{align*}
\]
\[ |L_n| = 1 + \max_{\text{all } j \text{ s.t. } S[j] < S[n]} |L_j| \]

Recursion: \( L_n \)

BAD

\( L_{n-1} \quad L_{n-2} \quad \ldots \quad L_1 \)

\( L_{n-2} \quad L_{n-3} \quad \ldots \quad \text{etc} \)

Dyn. Prog: Build solutions, "bottom up"

When it's time to solve \(|L_k|\) we have stored all \(|L_j| \quad (j < k)\) in an array.
$23, 3, 5, 18, 10, 101, 12, 14, 4$

$1$

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

**Recursion:** $L_n$

**BAD**

$L_{n-1}, L_{n-2}, \ldots, L_1$

$L_{n-2}, L_{n-3}, \ldots, \text{etc}$

**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.
23, 3, 5, 18, 10, 101, 12, 14, 4

1

$|L_2|$

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

Recursion: $L_n$

BAD

$L_{n-1}, L_{n-2}, \ldots, L_1$

$L_{n-2}, L_{n-3}, \ldots$

d 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.
$23, 3, 5, 18, 10, 101, 12, 14, 4$

$1 \quad 1^2$

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

Recursion: $L_n$

BAD $L_{n-1} \quad L_{n-2} \quad \ldots \quad L_1$

$\quad L_{n-2} \quad L_{n-3} \quad \ldots \quad \text{etc}$

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

Recursion: \(L_n\)

BAD

\(L_{n-1}, L_{n-2}, \ldots, L_1\)

\(L_{n-2}, L_{n-3}, \ldots\) 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.
\[ |L_n| = 1 + \max \{ \text{all } j \text{ s.t. } S[j] < S[n] \} |L_j| \]

**Recursion:** \( L_n \)

**BAD**

\( L_{n-1} \quad L_{n-2} \quad \ldots \quad L_1 \)

\( L_{n-2} \quad L_{n-3} \quad \ldots \quad \text{etc} \)

**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.
23, 3, 5, 18, 10, 101, 12, 14, 4
1 1 2 3 3 4

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

Recursion: \text{Ln}

BAD \quad L_n \quad L_{n-1} \quad L_{n-2} \quad \ldots \quad L_1

|\text{Ln-2} \quad \text{Ln-3} \quad \ldots \quad \text{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.
\[ |L_n| = 1 + \max \{ \text{all } j \text{ s.t. } S[j] < S[n] \} |L_j| \]

**Recursion:** \( L_n \)

**BAD**

\[ L_{n-1} L_{n-2} \ldots L_1 \]

\[ L_{n-2} L_{n-3} \ldots \text{ etc} \]

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

Recursion: \( L_n \)

BAD

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

\( L_{n-2} L_{n-3} \ldots \)

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

Recursion: \( L_n \)

BAD

\[ \ldots L_n-2 L_n-1 L_n \]

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.

Score may decrease

\[ 23, 3, 5, 18, 10, 101, 12, 14, 4 \]

1 → 2

2 → 3

3 → 4

4 → 5

5 → 2
\[
|L_n| = 1 + \max_{\{\text{all } j \text{ s.t. } S[j] < S[n]\}} |L_j|
\]

Recursion: \( L_n \)

BAD

\( L_{n-1} \quad L_{n-2} \quad \ldots \quad L_1 \quad L_{n-2} \quad L_{n-3} \quad \ldots \quad \text{etc} \)

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.

\(\text{time? space?}\)
Recursion: \( L_n \)

BAD

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

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 = \( \Theta(n) \)
\[ T(n) = \Theta(n^2) \]

Space = \( \Theta(n) \)

\[ |L_n| = 1 + \max_{\{\text{all } j \text{ s.t. } S[j] < S[n]\}} |L_j| \]
\[ T(n) = \Theta(n^2) \]

space = \Theta(n)

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

What about \(|L.I.S.| = |L_n|\)?
$23, 3, 5, 18, 10, 101, 12, 14, 4$

$1 \quad 1 \quad 2 \quad 3 \quad 3 \quad 4 \quad 4 \quad 5 \quad 2$

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

What about $|\text{L.I.S.}|$? = $|L_n|$? → NO.

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

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

space = \( \Theta(n) \)

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

What about \(|L.I.S.|\)?

\[ = |L_n|? \rightarrow \text{NO.} \quad \max_{j=1..n} |L_j| \]

What about L.I.S.?
\[ T(n) = \Theta(n^2) \]
\[ \text{space} = \Theta(n) \]

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

What about \( |L\text{-I.S.}| \)?

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.
A quick solution for L.I.S. ... but still $O(n^2)$ & dyn-prog.

23, 3, 5, 18, 10, 101, 12, 14 : S
A quick solution for L.I.S. ... but still $O(n^2)$ & dyn-prog.

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

sort $\leftarrow$

$$3, 5, 10, 12, 14, 18, 23, 101 : S_2$$

and then?
A quick solution for L.I.S. ... but still $O(n^2)$ & dyn-prog.

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

$3, 5, 10, 12, 14, 18, 23, 101 : S_2$

Find longest common subsequence!
A quick solution for L.I.S. ... but still $O(n^2)$ & dyn-prog.

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

$$3, 5, 10, 12, 14, 18, 23, 101 : S_2$$

**Find longest common subsequence!**

- any common subsequence is increasing (assume no duplicates) or remove them
A quick solution for L.I.S. ... but still $O(n^2)$ & dyn-prog.

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

Find longest common subsequence!

- Any common subsequence is increasing (assume no duplicates)
  - So $\text{LCS}(S, S_2)$ qualifies as a solution
A quick solution for L.I.S. ... but still $O(n^2)$ & dyn-prog.

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

sort ↓

$3, 5, 10, 12, 14, 18, 23, 101 : S_2$

**Find longest common subsequence!**

- any common subsequence is increasing (assume no duplicates)
  - so $\text{LCS}(S, S_2)$ qualifies as a solution
- $\text{LIS}$ must exist in $S_2$
A quick solution for L.I.S. ... but still $O(n^2)$ & dyn-prog.

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

\text{Find longest common subsequence!}

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