DYNAMIC PROGRAMMING – LONGEST INCREASING SUBSEQUENCE

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

$L(S) = 3, 5, 10, 12, 14, 105$ $\quad |L(S)| = 6$

Brute-force: try all subsequences, see if they are increasing: O(2^n)

For dynamic programming we would like

- a recursive expression w/ repeated subproblems
- an easy, fast way to use solved subproblems

$$\begin{array}{cccccccc} 1 & 2 & 3 & \dots & & \dots & n-1 & n \\ 23, & 3, & 5, & 18, & 10, & 101, & 12, & 14 \end{array}$$

$|L_n|$ using $|L_{n-1}|$?

$L(S) \sim$
$\sim L_{1..n}(S)$
$\sim L_n$

if $S[n] >$ last element in L_{n-1} then $|L_n| = |L_{n-1}| + 1$

↳ could be at any position in S

else // $S[n] \leq last(L_{n-1})$

→ keep $|L_{n-1}|$?

→ add $S[n]$ to suboptimal solution from $S[1...n-1]$?

$$\begin{array}{cccccccccc} 1 & 2 & 3 & \dots & & \dots & & & n-1 & n \\ 23, & 3, & 5, & 18, & 10, & 101, & 12, & 14, & 4, & 105 \end{array}$$

$|L_{n-1}| = 2$

Redefine: L_n = longest increasing subsequence that actually uses $S[n]$

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

look at all L_j $(j<n)$

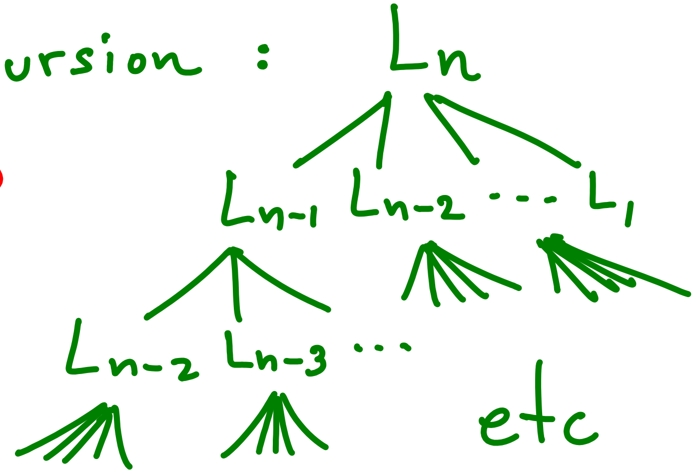
23 , 3 , 5 , 18 , 10 , 101 , 12 , 14 , 4

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 :

BAD



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 $= \Theta(n)$

23	3	5	18	10	101	12	14	4
1	1	2	3	3	4	4	5	2

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

space $= \Theta(n)$

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

What about $|L.I.S.|$? $\quad = \max_{j=1..n} |L_j|$

What about L.I.S. ? Keep the pointers: for each $S[j]$ store any $S[i]$ pointer that generated $|L_j|$

$\square$

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
  so $LCS(S, S_2)$ qualifies as a solution

- LIS must exist in S_2, so it is a candidate for LCS.

$\square$