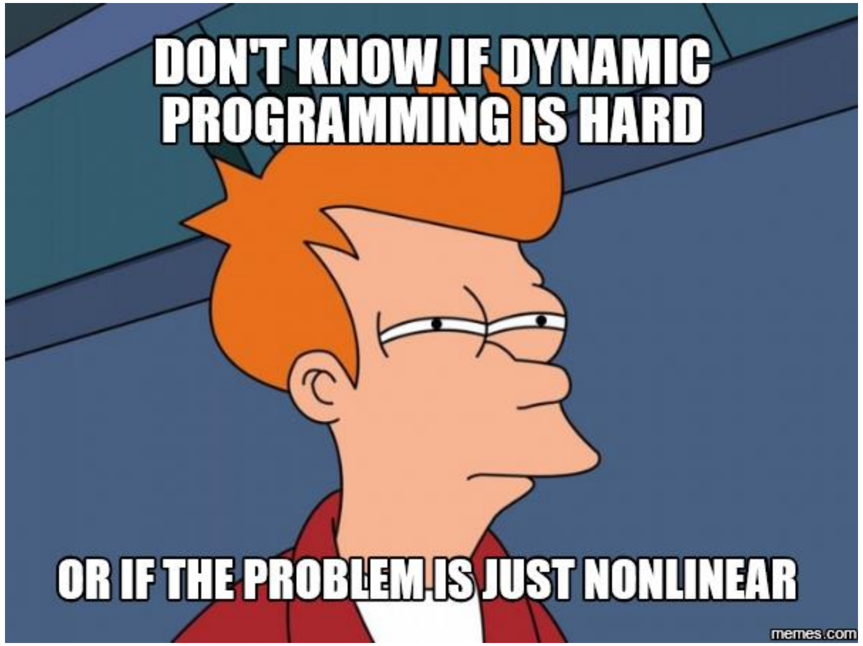
Extra Slides: LIS and LCS



Sec 101: MW 3:00-4:00pm DOW 1018
IA: Eric Khiu

Longest Increasing Subsequence

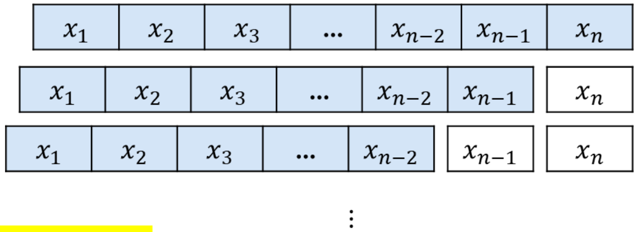
Notes

Longest Increasing Subsequence

- Given a sequence of numbers S, an increasing subsequence of S is a subsequence such that the elements are in strictly increasing order.
  - One LIS of $S = [0, 8, 4, 12, 5, 6, 3]$ would be $[0, 4, 5, 6]$
- Goal: Return the **length** of LIS of a given sequence
- **Discuss:** Why can't we solve this using divide and conquer?
  - Overlapping subproblem!

Longest Increasing Subsequence

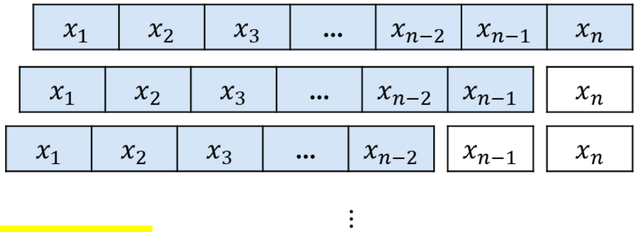
- **Step 0: Dimensionality**
  - 1-dimensional



- **Step 1: Subject of recurrence**
  - Attempt 1: $L[i]$ = Length of LIS on subarray $S[1, \ldots, i]$
  - Problem: The subsequence chosen can be *very* ambiguous
    - Example: In $[0, 8, 4, 12, 5]$, $L[5] = 3$, do we want $[0, 4, 5]$ or $[0, 8, 12]$?
    - Problematic because when we consider the next element, 6, we can only append it to $[0, 4, 5]$ and have $L[6] = 4$. If my $L[5]$ means $[0, 8, 12]$, then my $L[6]$ is still 3
    - **Reminder:** In DP, we don't want to re-solve the same subproblem again

Longest Increasing Subsequence

- **Step 0: Dimensionality**
  - 1-dimensional



- **Step 1: Subject of recurrence**
  - Attempt 2: $L(i)$ = Length LIS on subarray $S[1, \dots, i]$ that ends at $S[i]$
    - Using the same example: $[0, 8, 4, 12, 5]$, $L[5] = 3$
    - When we consider *any future* element at position k, we only have to check if it's greater than $L[i]$ to decide if we want to append $S[k]$ to that LIS ending at $S[i]$
- **Step 2: Base case(s)**
  - $i = 1$: $L[i] = 1$ (only one element)

Longest Increasing Subsequence

x_1	x_2	x_3	…	x_{n-2}	x_{n-1}	x_n
x_1	x_2	x_3	…	x_{n-2}	x_{n-1}	x_n
x_1	x_2	x_3	…	x_{n-2}	x_{n-1}	x_n

⋮

$L(i)$ = Length LIS on subarray $S[1,\ldots,i]$ that ends at $S[i]$

- **Step 3: Optimal Sub-solution**
  - (1) We can append $S[i]$ to *any* subsequence ending at $S[j], j < i$
  - (2) Only append if it is an increasing subsequence, i.e., $S[j] < S[i]$
  - [Optimal] Which subsequence to append to? The longest one! (across all j < i)

$$\max_{j:j<i}\{L(j): S[j] < S[i]\}$$

- Therefore, we have the recurrence relation

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ 1 + \max_{j:j<i}\{L(j): S[j] < S[i]\} & \text{otherwise} \end{cases}$$

or equivalently,

$$L(i) = 1 + \begin{cases} 0 & \text{if } i = 1 \\ \max_{j:j<i}\{L(j): S[j] < S[i]\} & \text{otherwise} \end{cases}$$

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ 1 + \max\limits_{j:j<i}\{L(j): S[j] < S[i]\} & \text{otherwise} \end{cases}$$

Longest Increasing Subsequence

- What's next? From cookbook:
  - To fill in cell, which other cells do I look at? L[j] for all j < i
  - Which cell(s) contain the final answer? $\max\limits_{i \in [1,n]} L[i]$

LIS($S[1, \dots, n]$):

  allocate $L[1, \dots, n]$

  $L[1] \leftarrow 1$

  for $i = 2, \dots, n$ **do**

    $L[i] \leftarrow 1 + \boxed{\max\limits_{j:j<i}\{L(j): S[j] < S[i]\}}$ We need a for loop here

  return $\max\limits_{i:1 \le i \le n} L[i]$

Q: What is the runtime of the algorithm?

$O(n^2)$

Longest Increasing Subsequence

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ 1 + \max\limits_{j:j<i}\{L(j): S[j] < S[i]\} & \text{otherwise} \end{cases}$$

LIS($S[1, \dots, n]$):

allocate $L[1, \dots, n]$

$L[1] \leftarrow 1$

for $i = 2, \dots, n$ **do**

$L[i] \leftarrow 1 + \max\limits_{j:j<i}\{L(j): S[j] < S[i]\}$

return $\max\limits_{i:1 \le i \le n} L[i]$

- Written more explicitly (good practice for actual coding)

LIS($S[1, \dots, n]$):

allocate $L[1, \dots, n]$

$L[1] \leftarrow 1$

for $i = 2, \dots, n$ **do**

$m \leftarrow 0$ // to keep track of max over j's

for $j = 1, \dots, i - 1$

if $S[j] < S[i]$ **and** $L[j] > l$ **then**

$m \leftarrow L[j]$

$L[i] \leftarrow 1 + m$

$l \leftarrow 0$

for $i = 1, \dots, n$ **do**

if $L[i] > l$ **then** $l \leftarrow L[i]$

return $\boldsymbol{l}$

Longest Common Subsequence

Notes

Longest Common Subsequence

- Given two strings A and B a common subsequence is a sequence that can be derived from both arrays in the same order without rearranging them.
- For example, $A =$" Go_Blue" and $B =$"Wolverines", a longest common subsequence is "ole"
- Goal: Return the **length** of LCS between the two strings

Longest Common Subsequence

- **Step 0: Dimensionality**
  - 2-dimensional: One "pointer" for each string
- **Step 1: Subject of Recurrence**
  - $L(i, j)$ = Length of LCS between $A[1, \dots, i]$ and $B[1, \dots, j]$
  - Unlike LIS, we do not enforce the subsequence to end at $A[i]$ nor $B[j]$
- **Step 2: Base Case(s)**
  - If $i = 0$ or $j = 0 \Rightarrow$ One of the string is empty $\Rightarrow L(i, j) = 0$

Subject of Recurrence
$L(i, j)$ = Length of LCS between $A[1, \dots, i]$ and $B[1, \dots, j]$

Longest Common Subsequence

- **Step 3: (Optimal) Sub-Solution**
  - When considering $L(i, j)$, we have two possibilities
  - (1) $A[i] = B[j]$ Hooray the common subsequence is now longer! +1 and recurse into $L(i-1, j-1)$
  - (2) $A[i] \neq B[j]$ **Discuss:** Why is it not a good idea to just recurse into $L(i-1, j-1)$ here?

    $A[i]$ or $B[j]$ might still be part of the LCS!
  - Two choices here: (a) Recurse into $L(i, j-1)$ or (b) Recurse into $L(i-1, j-1)$
  - Which one to choose? The *best* one!
  - [Optimal] Maximization problem: Use max

$$L(i, j) = \max\{L(i-1, j), L(i, j-1)\}$$

Subject of Recurrence
$L(i, j)$ = Length of LCS between $A[1, \dots, i]$ and $B[1, \dots, j]$

Longest Common Subsequence

- Putting everything together, we have the recurrence relation

$$L(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ 1 + L(i-1, j-1) & \text{if } A[i] = B[j] \\ \max\{L(i-1,j), L(i, j-1)\} & \text{otherwise} \end{cases}$$

- **Sanity check:** To fill in $L(i,j)$, which cells do I (potentially) need to look at?
  - $L(i-1, j-1), L(i-1, j), L(i, j-1)$
- Now, suppose we have the DP table for $L(i,j)$, which cell contains the final solution (length of LCS)?
  - $L(n,m)$: Length of LCS between $A[1, \dots, n]$ and $B[1, \dots, m]$

$$L(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ 1 + L(i-1, j-1) & \text{if } A[i] = B[j] \\ \max\{L(i-1,j), L(i,j-1)\} & \text{otherwise} \end{cases}$$

Longest Common Subsequence

LCS($A[1,\ldots,n]$, $B[1,\ldots,m]$):

Initialize an $(n+1) \times (m \times 1)$ table L

$L[0][j] \leftarrow 0$ **for** $j = 1,\ldots,m$

$L[i][0] \leftarrow 0$ **for** $i = 1,\ldots,n$

for $i = 1,\ldots,n$ **do**

for $j = 1,\ldots,m$ **do**

if $A[i] = B[j]$ **then** $L[i][j] \leftarrow 1 + L[i-1][j-1]$

else $L[i][j] \leftarrow \max\{L[i-1][j], L[i][j-1]\}$

return $L[n][m]$

		W	o	l	v	e	r	i	n	e	s
	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0
o	0	0	1	1	1	1	1	1	1	1	1
	0	0	1	1	1	1	1	1	1	1	1
b	0	0	1	1	1	1	1	1	1	1	1
l	0	0	1	2	2	2	2	2	2	2	2
u	0	0	1	2	2	2	2	2	2	2	2
e	0	0	1	2	2	3	3	3	3	3	3
!	0	0	1	2	2	3	3	3	3	3	3

Runtime: $O(nm)$

Space Complexity: $O(nm)$