Dynamic Programming

Tuesday, February 1, 2022 9:29 AM

"Sequencing DP"

Longest Increasing Subsequence (LIS)

Input: $A = [a_1, a_2, \dots, a_n]$ $a_i \in \mathbb{R}$.

Output: Length of the LIS.

Def: A subsequence is $a_{i_1}\, a_{i_2}\, a_{i_3} \dots a_{i_k}$

where $1 \le i_1 < i_2 < i_3 < \dots < i_k \le n$

Ex: $A = [2, 3, 1, -4, 1, 2, 6]$

$[2, 3, -4]$, $[2, 2]$, $[1]$ ✓

$[3, 1, 6, 2]$:(

Def: A subseq. is increasing if

$a_{i_1} < a_{i_2} < a_{i_3} < \dots < a_{i_k}$

1. Define a table describing your smaller problems.

$T[i]$ = the len of LIS of $[a_1, a_2 \dots a_i]$ ending at $\boxed{a_i}$

	0	2	1	-4	$\boxed{3}$	2	6
$T[i]$	1	2	2	1	$\boxed{3}$		

2. Write a recurrence to get the value of $T[i]$ from $T[j]$ $\underline{1 \le j < i}$.

$$T[i] = \max_{j<i} \{1 + T[j] \text{ if } a_j < a_i\}$$

Base case(s): $T[1] = 1$

3. Write pseudocode for your solution!

$LIS[a_1, \dots, a_n]$

$O(n^2)$
```
FOR i=1 to n:
    T[i]=1
    FOR j=1 to i-1:
        IF a_j < a_i AND T[i] < 1+T[j]:
            T[i] = 1 + T[j]
```

```
max=1
FOR i=2 to n:        <- O(n)
    IF T[i] > T[max]:
        max=i
RETURN T[max]
```

4. State and analyse the runtime (in big-O) of your design.

$O(n^2)$ because we have two nested loops of len $\boxed{n}$.

Longest Common Subsequence (LCS).

Input: $X = [AABDCA]$

$Y = [ABCDAA]$

Output: len of the LCS.

(WRONG)

$T[i]$ = len of the LCS out of $X = [x_1, x_2 \dots x_i]$, $Y = [y_1, y_2 \dots y_i]$

$T[1] = 1$

$T[2] = 1$

$T[3] = 2$