Thursday, February 29, 2024 9:18 AM

Last Lecture:

(LIS)

Problem: Longest Increasing Subsequence

Given a sequence of nos.

$a_1, a_2, \ldots, a_n$. $a_{n+1} = \infty$

Find $a_{i_1}, a_{i_2}, \ldots, a_{i_\ell}$ maximizing ℓ } opt. soln

s.t. $i_1 < i_2 < \ldots < i_\ell$

& $a_{i_1} < a_{i_2} < \ldots < a_{i_\ell}$

e.g. 8, 2, 3, 1, 10, 5, 17, 3, 9, 7, 12 } opt. sol'n $\ell = 5$

★ Recursive algo.

LIS ($\langle a_1, \ldots, a_n \rangle$, X) {

// Find "length" of LIS in $\langle a_1 \ldots a_n \rangle$

// s.t. max ele $< X$

1. If $n = 0$ return 0
2. If $a_n < X$ then do

return max { LIS ($\langle a_1, \ldots a_{n-1} \rangle$, X), $1 + LIS(\langle a_1, \ldots, a_{n-1} \rangle, a_n)$ }

3. else return LIS ($\langle a_1 \ldots a_{n-1} \rangle$, X) $\rightarrow T(n-1)$

}

- call LIS ($\langle a_1 \ldots a_n \rangle$, ∞) $a_{n+1} = \infty$
- Runtime: $T(n) = 2T(n-1) + O(1)$

$= \boxed{O(2^n)}$ Horrible.

★ distinct subproblem LIS($\langle a_1, \ldots, a_i \rangle$, $X = a_j$ for $i = 0 \ldots n$, $j = 1 \ldots n+1$)

" " " $= n * (n+1)$

$= O(n^2)$

Store & Reuse ≡ Memoization.

$L[i,j] = (\langle a_1, \ldots, a_i \rangle, X = a_j)$ $a_{n+1} = \infty$

$0 \le i \le n$ $j = 1 \ldots n+1$

mark $L[i,j]$ = undef $\forall i, \forall j$

LIS (i, j) {

// return length of LIS in $\langle a_1 \ldots a_i \rangle$

s.t. max ele $< a_j$

1. $L[i,j] \neq$ undef then return $L[i,j]$

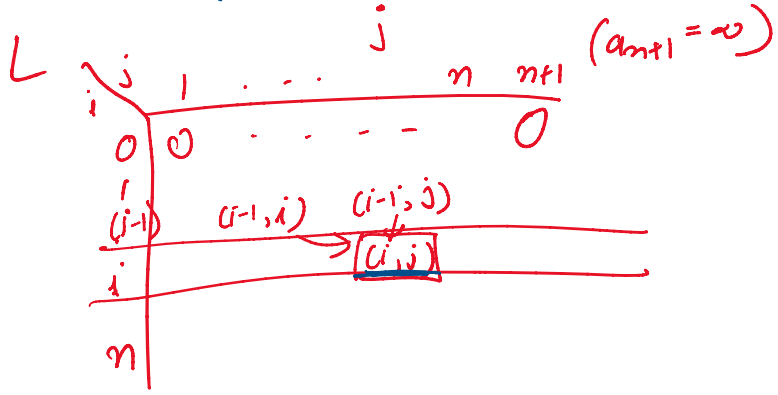
s.t. max ele $< a_j$

1. $L[i,j]$ + undef then return $L[i,j]$
2. If $i=0$ return $L[0,j] = 0$
3. If $a_i < a_j (=x)$

   then return $L[i,j] = \max\{LIS(i-1,j),\ 1 + LIS(i-1,i)\}$
4. else return $L[i,j] = LIS(i-1,j)$

}

Call: $LIS(n, n+1)$



Evaluation order:

increasing order of i.

Iterative Algo:

$O(n)$ 1. for $j = 1$ to $n+1$ $L[0,j] = 0$

2. for $i = 1$ to n {
3. for $j = 1$ to $n+1$ {
4. if $a_i < a_j$ then

   $L[i,j] = \max\{L[i-1,j],\ 1 + L[i-1,i]\}$

   else $L[i,j] = L[i-1,j]$

5. }
6. }

$O(n^2)$ $O(n)$ $O(1)$

Runtime: $O(n^2)$

Runtime: $O(n^2)$

Dynamic Programming:

- Define subproblems.
- Recursive formula
- Evaluation order.

Problem: Longest Common Subseq.

Given two seq $A = a_1 \ldots a_n$
$B = b_1 \ldots b_m$

Find longest subseq. of A that is also a subseq of B.

eg. ALGORITHM L — sol'n: LGRITHM
LOGARITHM G — sol'n: LORITHM
opt sol'n: 7

(Rmk: edit distance # insertion/deletions
UNIX diff)

☆ Subproblems: $0 \le i \le n$, $0 \le j \le m$

$C(i,j)$: LCS$(\langle a_1 \ldots a_i \rangle, \langle b_1, \ldots b_j \rangle)$

opt sol'n $C(n,m)$.

Intuition:
if $a_n = b_m$ then
opt sol'n = LCS$(\langle a_1, \ldots a_{n-1} \rangle, \langle b_1 \ldots, b_{m-1} \rangle)$

☆ Recursion:

Base case: either $i=0$ or $j=0$

$C[0,j] = C(i,0) = 0$

- Inductive cases for a_i & b_j

Induction: cases for a_i & b_j

$C(i,j)$

① drop a_i $\quad C(i-1, j)$

② drop b_j $\quad C(i, j-1)$

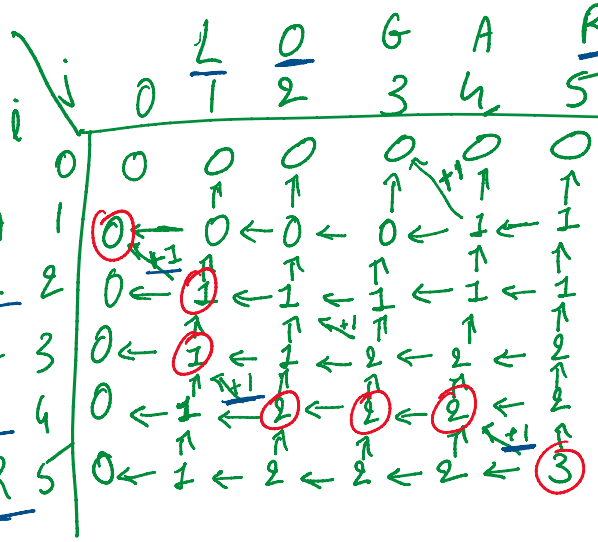
③ if $a_i = b_j$ then

1 + $C(i-1, j-1)$

$$C(i,j) = \max\{C(i-1,j), C(i,j-1)\} \quad \text{if } a_i \neq b_j$$

$$= \max\{C(i-1,j), C(i,j-1), 1 + C(i-1,j-1)\} \quad \text{if } a_i = b_j$$

$C[i,j]$



soln

LGR

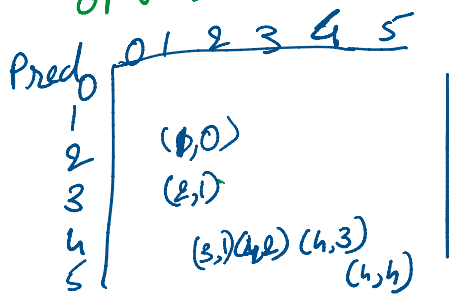
LOR

eg.

ALGOR

LOGAR

opt soln = 3



soln $C[n,m]$

Evaluation Order:

in increasing order of i

for each i, increasing order of j

Iterative Algo:

O(n) 1. for $i = 0$ to n $\quad c[i,0] = 0$

O(m) 2. for $j = 0$ to m $\quad c[0,j] = 0$

3. for $i = 1$ to n do

O(mn) { 4. for $j = 1$ to m do

O(m) { 5. if $a_i \neq b_j$ then

6. $c[i,j] = \max\{c[i-1,j], c[i,j-1]\}$

O(1) { else

7. $c[i,j] = \max\{c[i-1,j], c[i,j-1], 1 + c[i-1,j-1]\}$

8. }

}

9. Return $c[n,m]$.

}

$O(mn)$ Hence