Dynamic Programming

Weighted interval scheduling

Grid paths with obstructions

Problems over words/sequences

Word - sequence of letters from a fixed finite alphabet

A Σ

Given u, v words over A

Longest "overlap" - common subword

secret

bisect

Longest Common Subword

3 ← (length of)

$u = a_1 a_2 \dots a_n$

$v = b_1 b_2 \dots b_m$

Naive solution:

Best soln starts at some (a_i, b_j)

$a_i - -$
$b_j - -$ | count length of common word starting at (a_i, b_j)

Assume $n < m$ $\rightarrow O\left((n \cdot m) \cdot n\right)$ worst case

$n = m \Rightarrow O(n^3)$

$u = a_1 a_2 \cdots a_n$ ■

$v = b_1 b_2 \cdots b_m$ ■

$LCW(i,j)$ length of longest common subword from (a_i, b_j)

Inductive expression for $LCW(i,j)$

$a_i\, a_{i+1} \ldots a_n$

$b_j\, b_{j+1} \ldots b_m$

$a_i \neq b_j$: $LCW(i,j) = 0$

$a_i = b_j$: $LCW(i,j) = 1 + LCW(i+1, j+1)$

For convenience $LCW(n+1, j) = LCW(i, m+1) = 0$

Dependency $\quad LCW(i,j)$ depends on $LCW(i+1, j+1)$

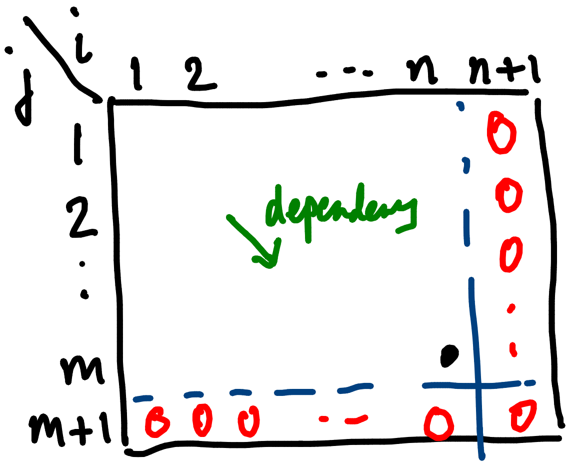
Base Cases:

$$\left.\begin{array}{l} LCW(n+1, j) = 0 \\ LCW(i, m+1) = 0 \end{array}\right\}$$

"Dummy" values are base cases

$$LCW(n,m) \begin{cases} 0 & \text{if } a_n \neq b_m \\ 1+0 & \text{if } a_n = b_m \end{cases}$$

— not required to treat as a base case



Can fill table

Recovering the actual word?

	b	i	s	e	c	t	
s	0	0	3	0	0	0	0
e	0	0	0	2	0	0	0
c	0	0	0	0	1	0	0
r	0	0	0	0	0	0	0
e	0	0	0	1	0	0	0
t	0	0	0	0	0	1	0
	0	0	0	0	0	0	0

- $O(m \cdot n)$ time
- Keep track of max value as we fill table : $LCW(3,1) = 3$
- Recover LCW by reading off 3 letters from a_3 (or b_1)

Generalize the problem

b i s e c t i o n

d i r e c t o r

Longest common subsequence - can drop some letters

e.g. ito is a common subsequence

ieto

iecto $\rightsquigarrow$ 5

a d b c a

b e a f b

Witnesses not unique

Focus on length of LCS first

longest common subsequence

Motivation :

- Similarity of DNA across species
- Computing discrepancies in text files   diff

LCS(i,j) – length of longest common subsequence from a_i, b_j

$a_i \; a_{i+1} \; \ldots \; a_n$

$b_j \; b_{j+1} \; \ldots \; b_m$

$a_i = b_j$: $1 + LCS(i+1, j+1)$ – Why?

$a_i \; \ldots \; a_{i+k} \; \ldots$

$b_j \; \ldots$

better soln?

Replace (a_{i+k}, b_j) by (a_i, b_j) – same quality soln

$a_i \; a_{i+1} \cdots a_n$

$b_j \; b_{j+1} \cdots b_m$

$a_i \neq b_j$

Both a_i & b_j cannot simultaneously be part of LCS

Either a_i or b_j could be there, but they exclude each other

If a_i is present. $LCS(i, j+1)$

b_j " " $LCS(i+1, j)$

Take max

As before, extend to $n+1$, $m+1$

$$LCS(n+1, j) = 0$$

$$LCS(i, m+1) = 0$$

$$LCS(i,j) = \begin{cases} 1 + LCS(i+1, j+1) & \text{if } a_i = b_j \\ \max\big(LCS(i, j+1), \\ \quad LCS(i+1, j)\big) & \text{if } a_i \neq b_j \end{cases}$$

	s	i	r	e	n	
f	3	3	2	2	1	0
i	3	3	2	2	1	0
e	2	2	2	2	1	0
n	1	1	1	1	1	0
d	0	0	0	0	0	0
	0	0	0	0	0	0

Dependencies?

$$\begin{tikzcd} LCS(i,j) \arrow[r] \arrow[d] \arrow[dr] & LCS(i+1,j) \\ LCS(i,j+1) & LCS(i+1,j+1) \end{tikzcd}$$

Recover answer by remembering choices

Just scan top row/first col & check each index where value drops?