DYNAMIC PROG. (DP)

Longest Common Subsequence (LCS) Problem

Given 2 strings $a = a_1 a_2 \cdots a_m$
$b = b_1 b_2 \cdots b_n$

find longest string that is a subseq of both a & b.

eg. → 0 1 0 1 3 1 2 1
a → 1 0 3 2 0 1 0 2
b → ↑ ↑ ↑ ↑ ↑

1 0 3 2 1

1 3 2 ~~1 3 2~~

(appl- comparing 2 strings / files / DNAs
min # deletes/inserts of single chars)

Define subproblems: for $i = 0, .., m$, $j = 0, .., n$

let $C(i,j)$ = length of LCS of $a_1 a_2 \cdots a_i$ & $b_1 b_2 \cdots b_j$

Want $C(m,n)$.

Recursive formula:

Case 1: soln not use a_i ⇒ $C(i,j) = C(i-1, j)$
Case 2: soln not use b_j ⇒ $C(i,j) = C(i, j-1)$
Case 3: Soln uses a_i and b_j with $a_i = b_j$.
⇒ $C(i,j) = C(i-1, j-1) + 1$

don't know which case, so try all & take max

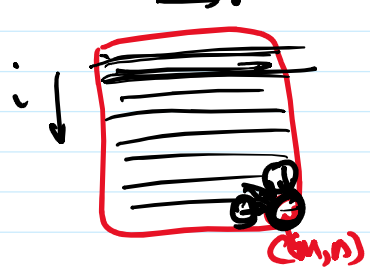
$C(i,j) = \max\{C(i-1,j), C(i,j-1), C(i-1,j-1)+1\}$

$$\Rightarrow C[i,j] = \begin{cases} \max\{C(i-1,j), C(i,j-1), C(i-1,j-1)+1\} & \text{if } a_i = b_j \\ \max\{C(i-1,j), C(i,j-1)\} & \text{if } a_i \neq b_j \end{cases}$$

Know: $C(i-1,j) \leq C(i-1,j-1)+1$, $C(i,j-1) \leq C(i-1,j-1)+1$

Base case: $C(i,0) = 0, \quad C(0,j) = 0$

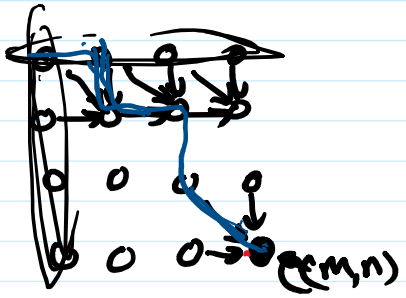
Evaluation order: increas. i
for same i, increas. j



Analysis: $O(mn)$ table entries
each in $O(1)$ time
$\Rightarrow$ $O(mn)$ time

$O(mn)$ space

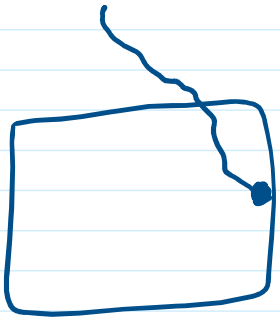
Rmk: can improve space to $O(n)$ ← but only if we just want opt value but not opt soln
by remembering only last 2 rows



How to output opt soln?

```
output-sol(i,j):
    if i=0 or j=0 return
    if C[i,j] = C[i-1,j-1]+1 & a_i=b_j
        output-sol(i-1, j-1), output a_i
    else if C[i,j] = C[i-1,j]
        output-sol(i-1,j)
    else output-sol(i,j-1)
```

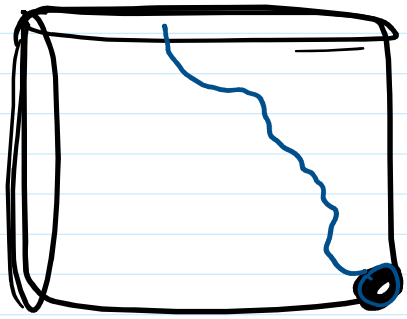
$O(m+n)$ time

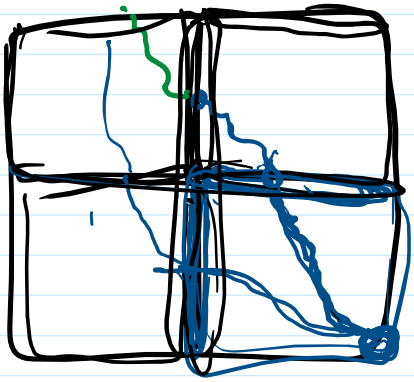


Space Improvement (Hirschberg '75 / Chowdhary, Ramachandran '06)

Suppose $m = n$

suppose $m = n$.





idea - divide & conquer

$$T(n) = \underbrace{O(n^2)}_{DP} + \cancel{4}\,3\,T\left(\frac{n}{2}\right) \Rightarrow O(n^2 \log n)$$

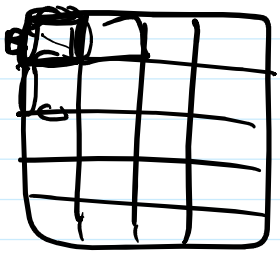
$O(n^2)$ time

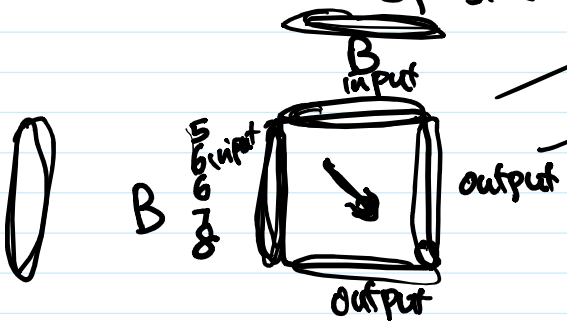
$$S(n) = O(n) + S\left(\frac{n}{2}\right) \Rightarrow O(n) \text{ space}$$

Time Improvement (Masek-Paterson '80)

Assume $|\Sigma|$ const. $\quad m \approx n$.

idea - precompute answers for all possible subproblems of size $B \times B$ & store in table $\left(\frac{n}{B}\right)^2$





$C[i,j]$

subproblems

$$\leq |\Sigma|^B \cdot |\Sigma|^B \cdot 2^B \cdot 2^B \quad \text{by storing diffs}$$

$$\leq (2|\Sigma|)^{2B} \leq |\Sigma|^{4B}$$

$\leq (2|\Sigma|)^{\varepsilon \log_{|\Sigma|} n} \leq |\Sigma|^{\varepsilon \log_{|\Sigma|} n} \leq n^{4\varepsilon}$

Set $B = \varepsilon \log_{|\Sigma|} n$

table preprocessing time $O(n^{4\varepsilon}(\log n)) \leq o(n)$

set $\varepsilon = 0.1$

$\Rightarrow$ total time $O\left(\left(\frac{n}{B}\right)^2\right) = \boxed{O\left(\frac{n^2}{\log^2 n}\right)}$

Rmk - known as "4 Russians trick"