Algs. based on dyn. progr.

Recap. of LCS :

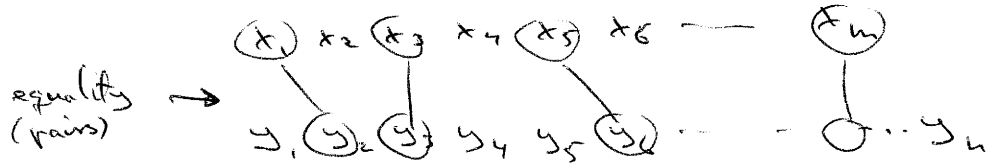
[String similarity
Unix diff.
Bioinf.]

- Define problem :

x string of length m
y " " n

subseq. : $x_1\ (x_2)\ (x_3)\ x_4\ (x_5) \cdots x_m$

common subseq :



longest C.S.

+ length
+ actual seq.

$LCS(i,j)$ = length of lcs of $x[1..i]$ and $y[1..j]$

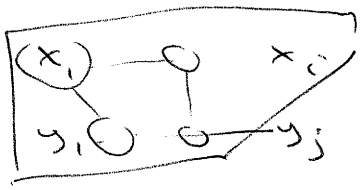
Dynamic Programming ~ build larger solutions from smaller

Lemma :

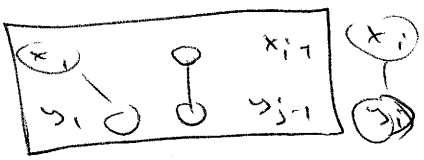
$$LCS(i,j) = \begin{cases} 0 & \text{if } i=0 \text{ OR } j=0 \quad i) \\ \max\{LCS(i-1,j), LCS(i,j-1)\} & \text{if } i,j \geq 1,\ x[i] \neq y[j] \quad ii) \\ LCS(i-1,j-1)+1 & \text{if } i,j \geq 1 \quad x[i]=y[j] \quad iii) \end{cases}$$

Proof:

Consider an opt-sol. and its pair(s):



ii) : (x_i, y_j) cannot be a pair.

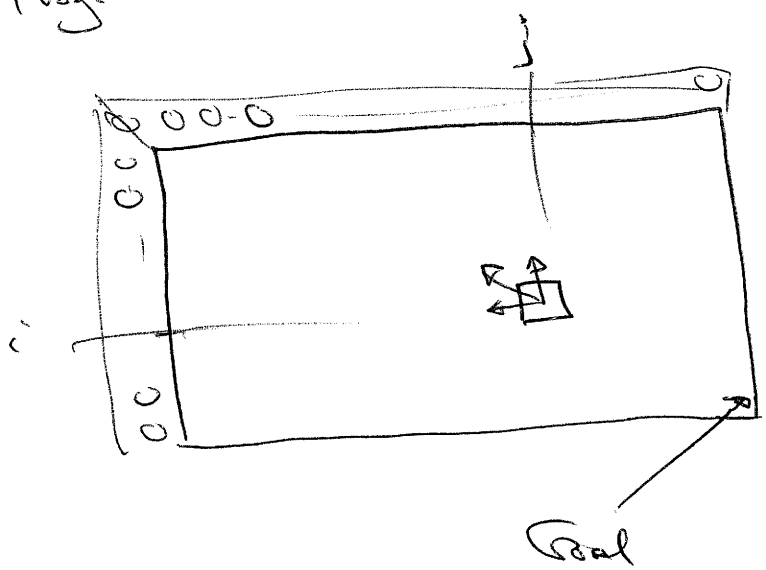


iii) Wlog (x_i, y_j) is pair

(i) is clear [one string is empty)

□

Dyn. Prog.:



Build by: → or ↓ (or ↘)

For finding LCS(m,n) :

time $O(m \cdot n)$

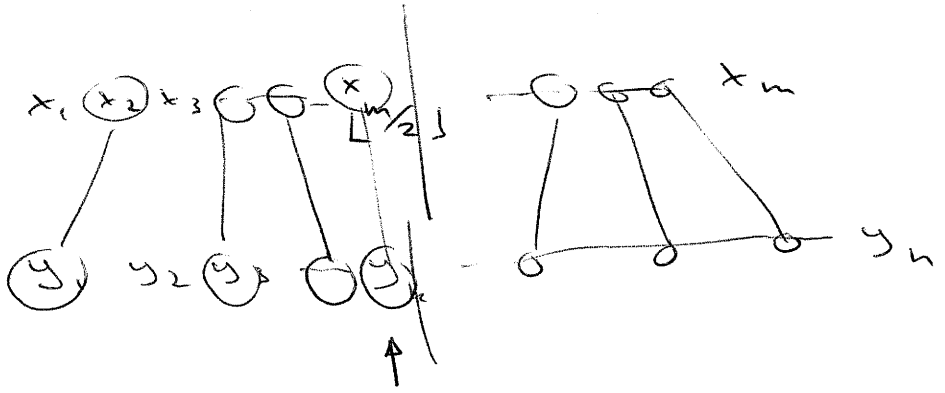
space $O(\min(m,n))$

*) For finding a CS with length LCS(m,n) :
Backtrack. Requires entire table stored $\Rightarrow$
$\Theta(m \cdot n)$ space (and same time)

Hirschberg trick for getting $O(\min(m,n))$ space for *) : [1975]

<u>W.l.o.g. $m \geq n$</u>

Look at an longest c.s. :



k = last index of char in y paired with char in $x[1 .. \lfloor m/2 \rfloor]$

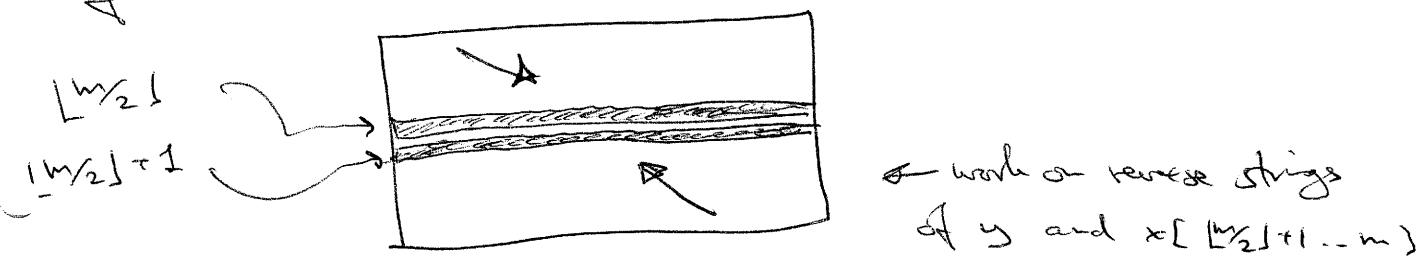
**) So $LCS(n,m) = \max_k \{ LCS(x[1..\lfloor m/2 \rfloor], y[1..k]) + LCS(x[\lfloor m/2 \rfloor + 1 .. m], y[k+1..n]) \}$

(as lengths and as subsequences)

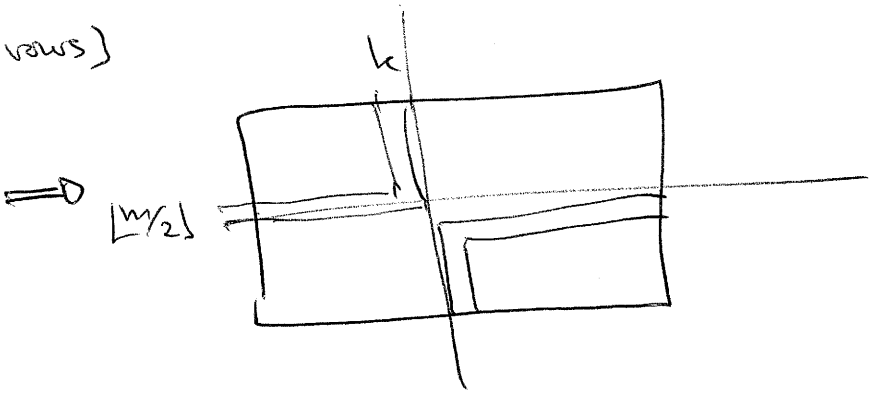
(Proof: $\leq$ + $\geq$).

Observation: LCS of strings a and b is equal to LCS of both strings reversed.

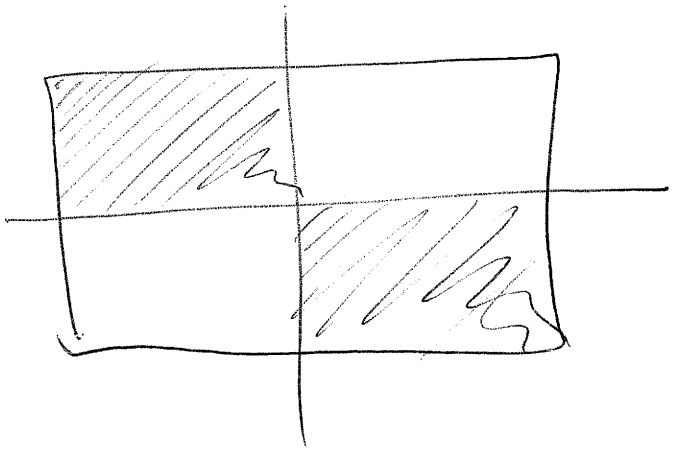
So we can in $O(m \cdot n)$ time and $O(n)$ space find dark rows in:



So max-arg k in **) can be found in same time [scan dark rows]

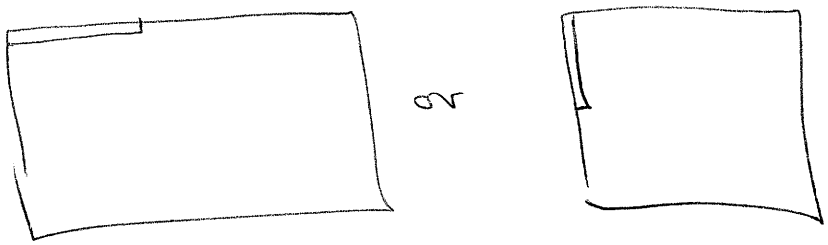


Recurse on



to find first and second half of ~~seq~~ LCS.

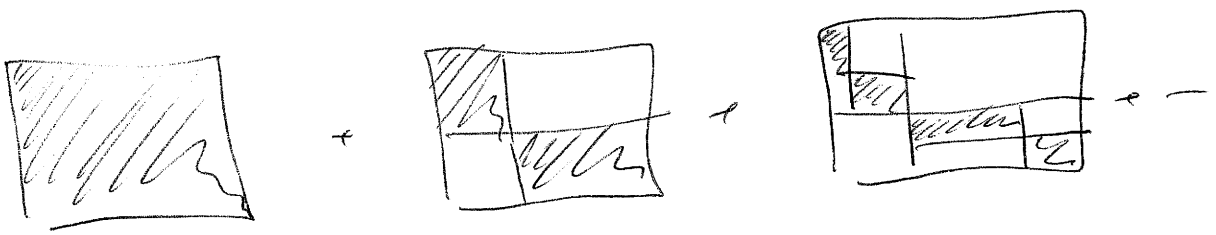
Stop criterion:



(or empty/zero in y-dimension (1st axis))

Easy: scan for existence of ~~[illegible]~~ char in single-~~[illegible]~~ char string in other string (or repeat empty c.s.)

Work: In each call prop. to "area covered".



Clearly = work of 1st call $\times \sum_{i=0}^{\infty} (\frac{1}{2})^i$

$= \_\_\_\_\_ \times O(1) = \Theta(m \cdot n)$

Space : $O(n)$ + $O(1)$ · (recursion stack)

(+ output)

$= O(n + \log n)$

= linear.