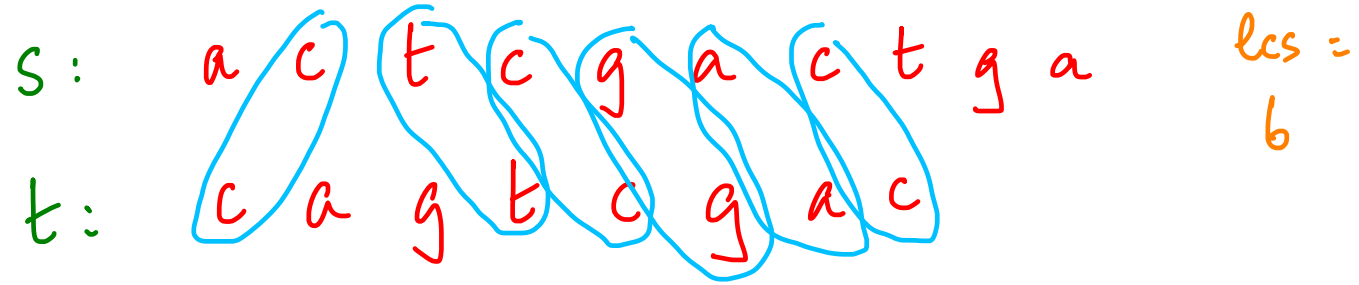
# Dynamic Programming

## Longest common subsequence (length of)




$s[1..n]$ $s_1 s_2 \cdots s_n$

$t[1..m]$ $t_1 t_2 \cdots t_m$

$lcs(i,j)$ solution for $s[1..i]$ $t[1..j]$

Base Case

$$lcs(1,1) = \begin{cases} 1 & \text{if } s_1 = t_1 \\ 0 & \text{otherwise} \end{cases}$$

Useful to add

$$lcs(0,j) = lcs(i,0) = 0 \quad \text{for } 0 \le i \le n,\ 0 \le j \le m$$

Inductive case

$lcs(i,j)$

$s_1\ s_2\ \ldots\ s_{i-1}\ s_i\ s_{i+1}\ \ldots\ s_n$

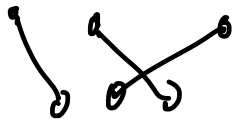
$t_1\ t_2\ \ldots\ t_{j-1}\ t_j\ t_{j+1}\ \ldots\ t_m$

If $s_i = t_j$, should we match them?

$s_1\ s_2\ s_3\ \ldots\ s_{i-1}\ s_i$

$t_1\ t_2\ t_3\ \ldots\ t_{j-1}\ t_j$

Matches do not "cross"



X

Assume $s_i = t_j$

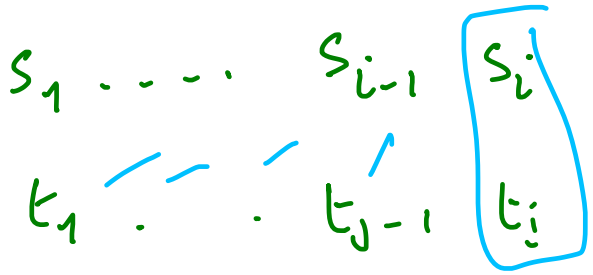
Optimal soln does not match them

Case 1 Neither $s_i$ nor $t_j$ is matched in optimal

Case 2/3 Exactly one of $s_i, t_j$ is matched

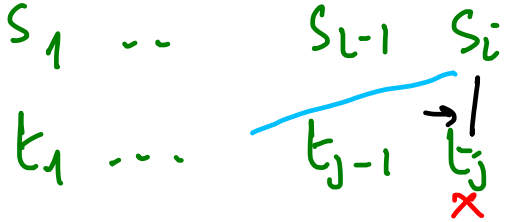
Case 4 Both matched, but not to each other

**Case 1** Neither $s_i$ nor $t_j$ appear in optimal

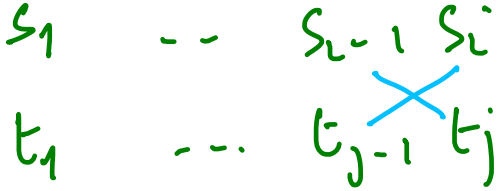



Earlier coda not optimal
- add $(s_i, t_j)$

**Case 2/3** Exactly one is matched




shift $s_i$ match to $t_j$
and preserve optimality

**Case 4** 


Not possible

$s_1 \ldots s_{i-1} s_i$

$t_1 \ldots t_{j-1} t_j$

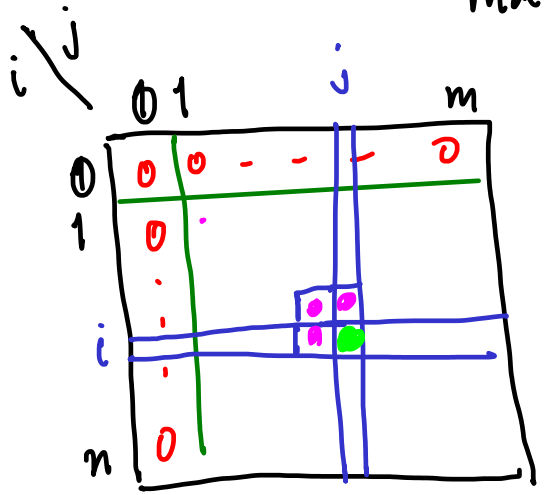
lcs(i,j) : if $s_i = t_j$ :

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

else: At least one of $s_i, t_j$ is useless

$max(lcs(i, j-1), lcs(i-1, j))$

lcs:




Like grid path,

fill by diagonal ////

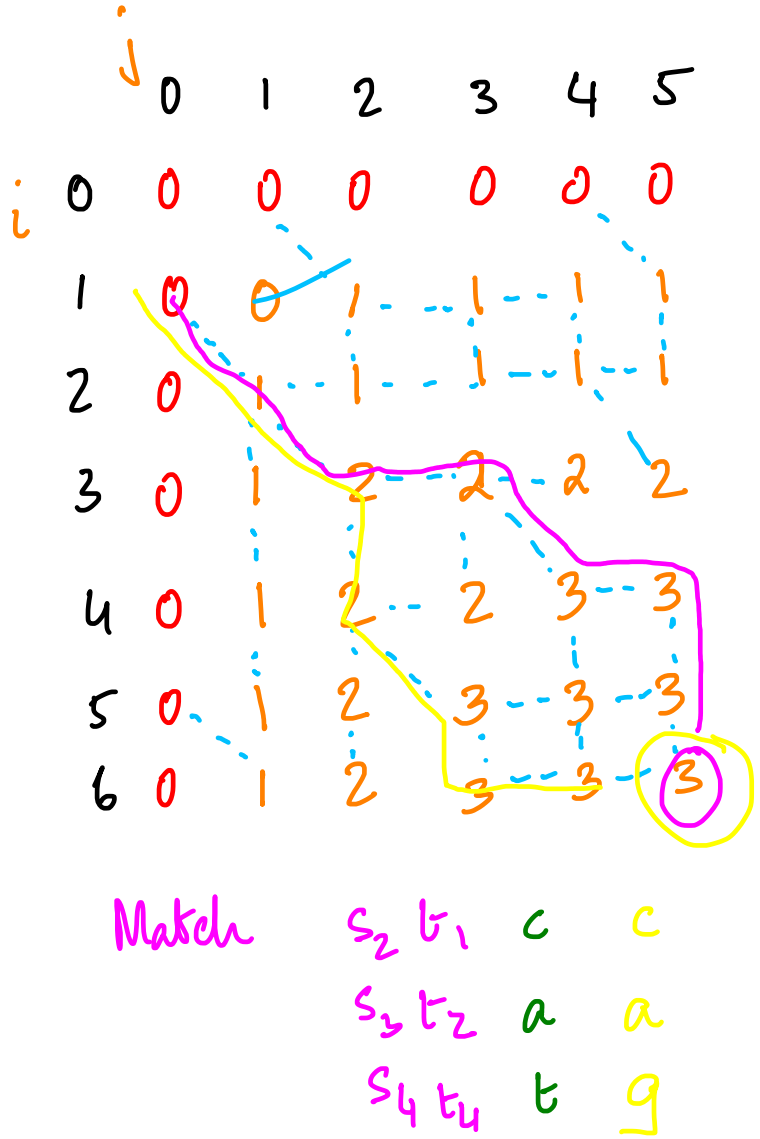
or by colum ↓↓↓↓

or by row →→→

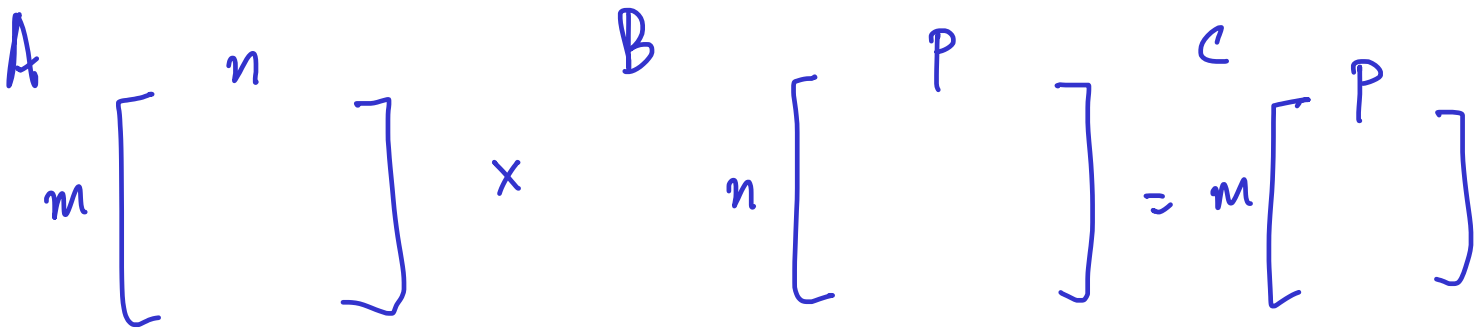
s: a c a t g c

t: ~~c~~ ~~a~~ ~~g~~ ~~t~~ a

Can recover the actual matched sequence (all of them!) by recording how each entry in table was calculated



# Optimal matrix multiplication

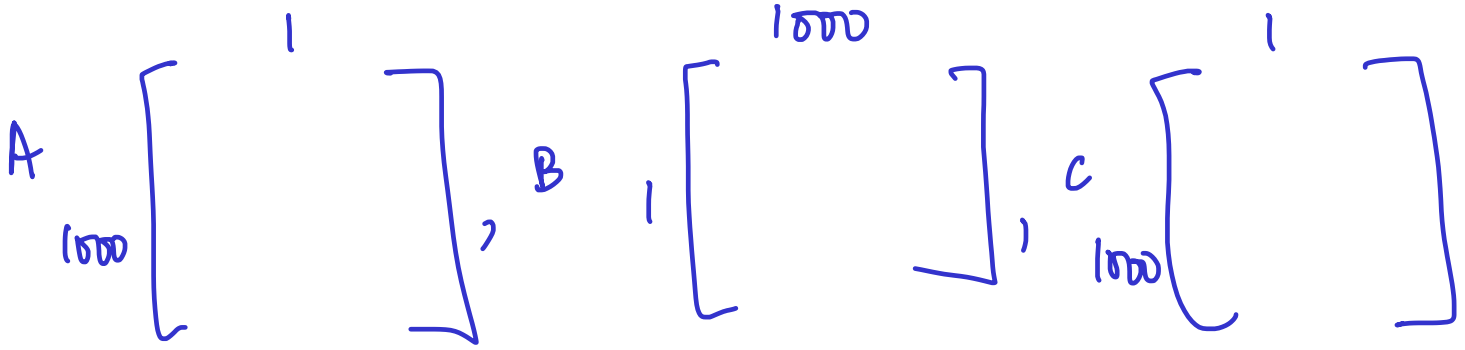



Using straightforward calculation of C

Each entry in C requires n multiply + add = O(n)

m·p entries ⇒ O(mnp)

Given




Compute $A \times B \times C$

$(A \times B) \times C$

$1000 \times 1 \times 1000$

$\downarrow$

$1000 \times 1000$

$1000 \times 1000$

$A \times (B \times C)$ ⇐ More efficient

$\downarrow$

$1 \times 1000 \times 1$

$1 \times 1$

$1000 \times 1 \times 1$

Given $M_1, M_2, \ldots M_n$

dimensions $(r_1, c_1), (r_2, c_2), \ldots, (r_n, c_n)$ $\quad c_i = r_{i+1}$

Find the optimal way to compute $M_1 \times M_2 \times \ldots \times M_n$

Best bracketing

Last step: $(M_1 \times \ldots \times M_j) \times (M_{j+1} \ldots \times M_n)$

$r_1 \qquad c_j = r_{j+1} \qquad c_n$

Cost: $r_1 \cdot c_j \cdot c_n +$ cost $(M_1 \ldots M_j)$ + cost $(M_{j+1} \ldots M_n)$

But which $j$ to choose?

$$Cost(1..n) = \min_{1 \le j \le n-1} r_1 \cdot c_j \cdot c_n + Cost(1..j) + Cost(j+1..n)$$

$$Cost(k..l) = \min_{k \le j \le l-1} r_k \cdot c_j \cdot c_l + Cost(k..j) + Cost(j+1..l)$$

$M_k \times M_{k+1} \times \ldots \times M_l$

Base Case:

$Cost(k..l) = 0$ if $k = l$

or even $k \ge l$

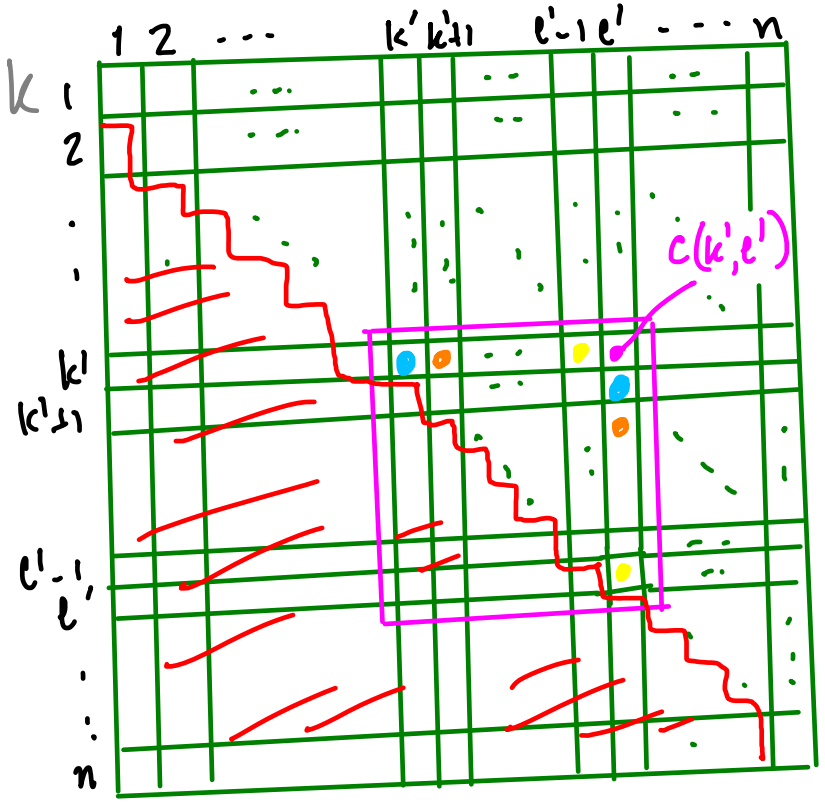
$M_1 \times M_2 \ldots \times (M_k \times M_{k+1} \ldots \times M_{l-1} \times M_l) \times M_{l+1} \times \ldots \times M_n$

# Table

Cost

Find a good order to fill this table

Only entries above diagonal : $k' \leq \ell'$




$c(k', \ell')$

$j$ varies from $k'$ to $\ell'-1$

✓ $c(k',k')\ c(k'+1,\ell')$

✓ $c(k',k'+1)\ c(k'+2,\ell')$

⋮

✓ $c(k',\ell'-1)\ c(\ell',\ell')$