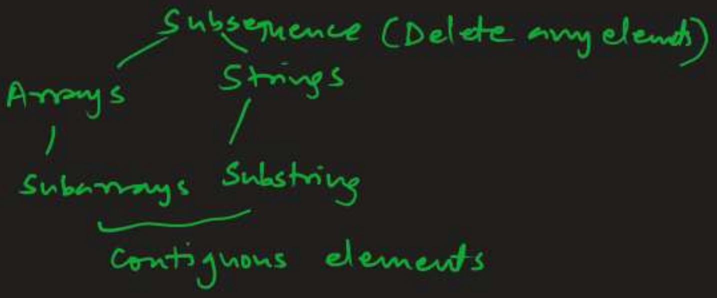
Dynamic Programming Lecture 2

Monday, 26 August 2024 6:03 AM

Longest Common Subsequence

↳ Given two sequences, the task is to find the longest subsequence that is common to both sequences.

↳ subsequence → sequence derived from another sequence by deleting zero or more elements without changing the order of the remaining elements.



S1 = A B A C D A E

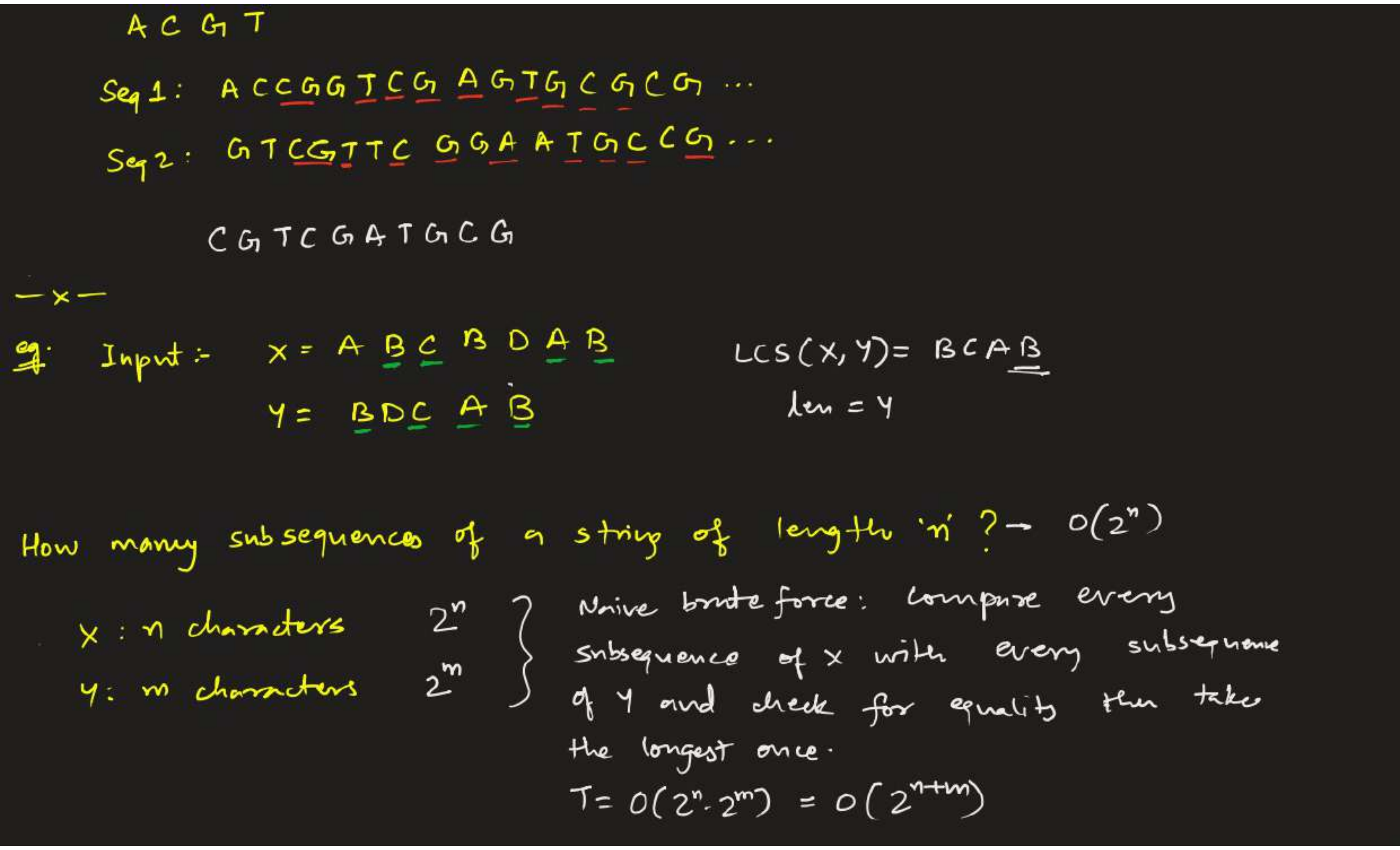
✓ S2 = A B A C D A E

✓ S3 = A

✓ S4 = A A A

✓ S5 = A B C D E

✗ S6 = A B C E A



A C G T

Seq 1: A C C G G T C G A G T G C G C G ...

Seq 2: G T C G T T C G G A A T G C C G ...

C G T C G A T G C G

—x—

eg: Input :- X = A B C B D A B

Y = B D C A B

LCS(X, Y) = BCAB

len = 4

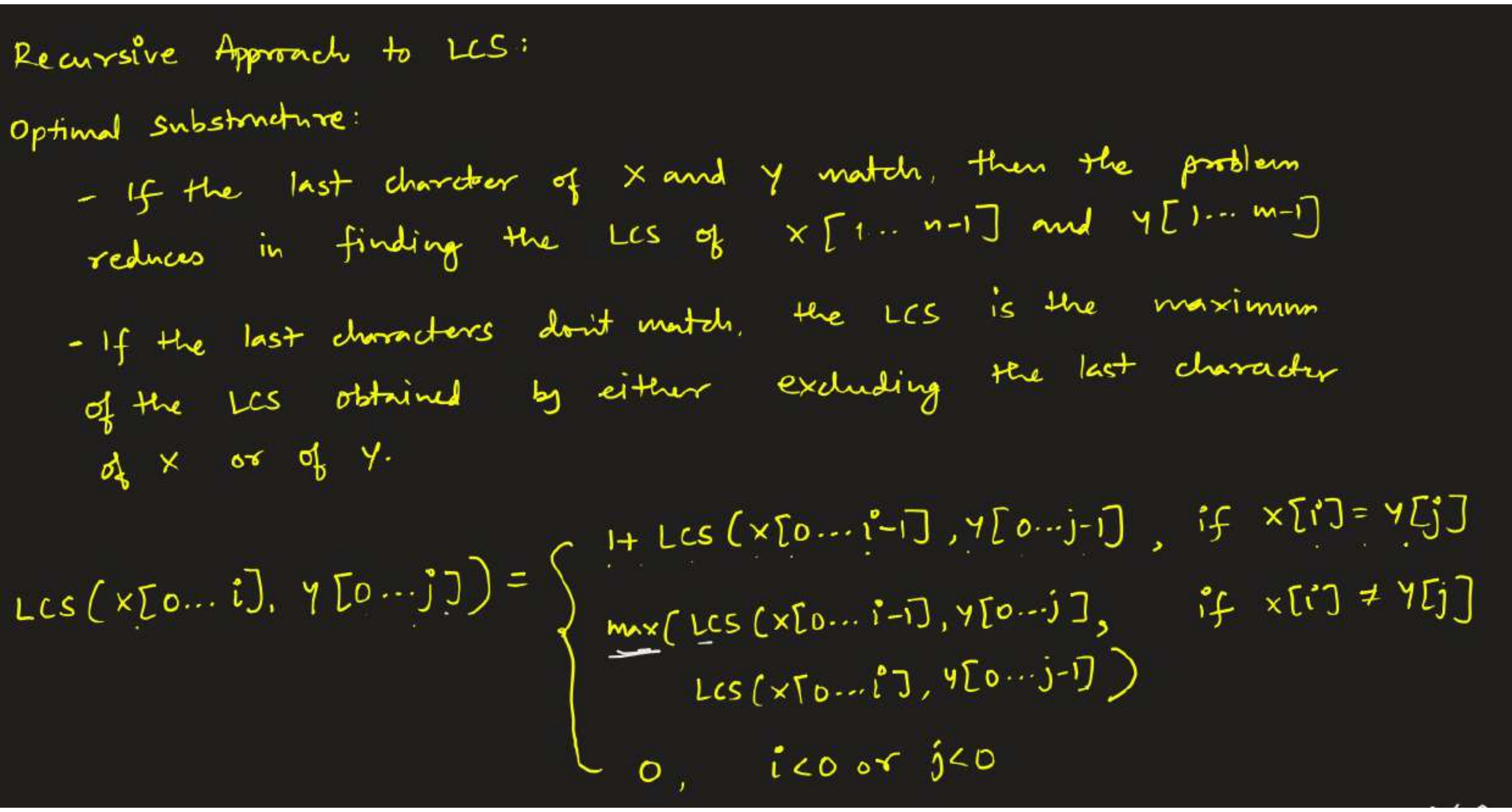
How many subsequences of a string of length 'n' ? → $O(2^n)$

X : n characters 2^n

Y : m characters 2^m

Naive brute force: compare every subsequence of X with every subsequence of Y and check for equality then take the longest once.

$T = O(2^n \cdot 2^m) = O(2^{n+m})$

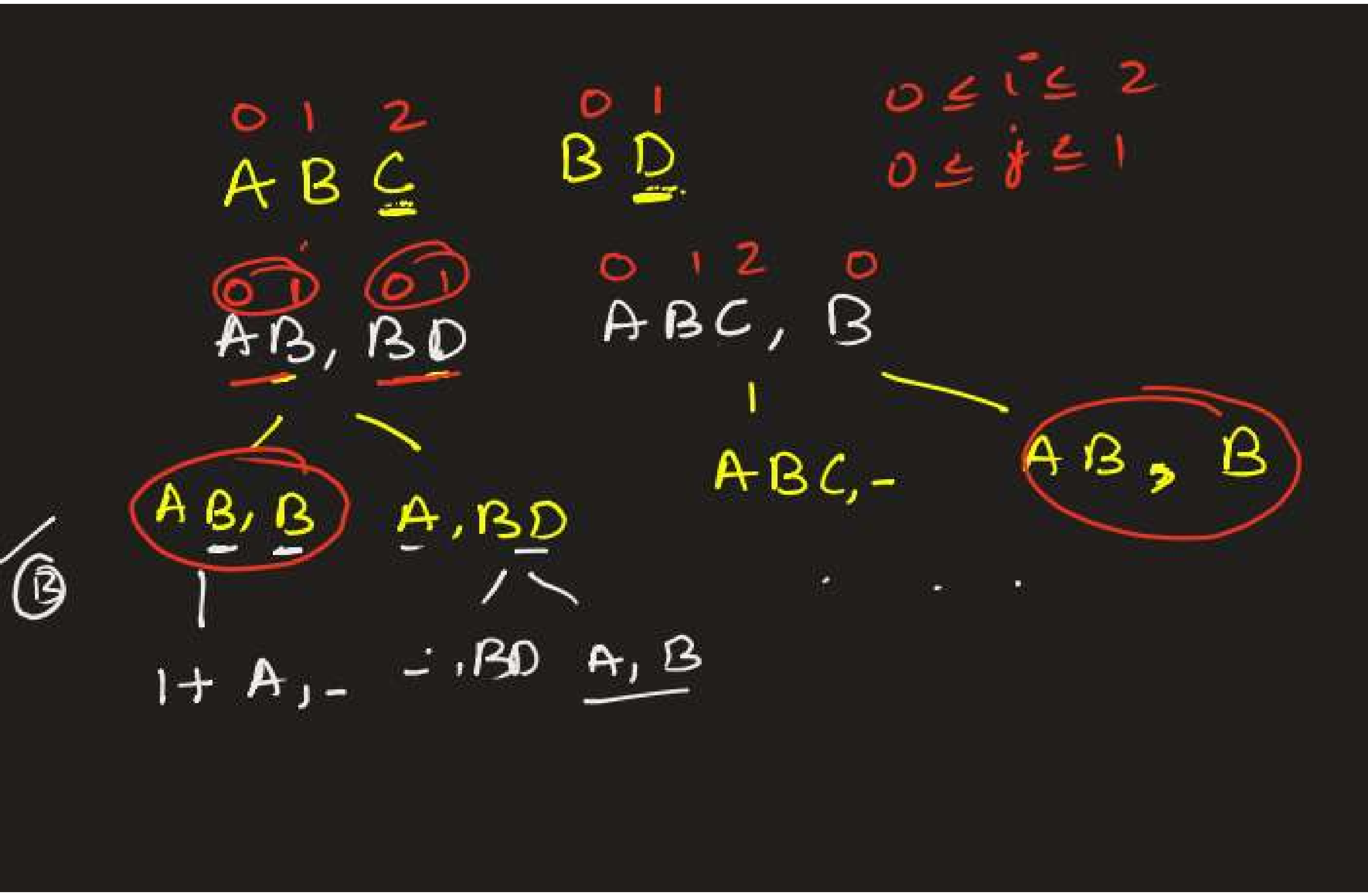


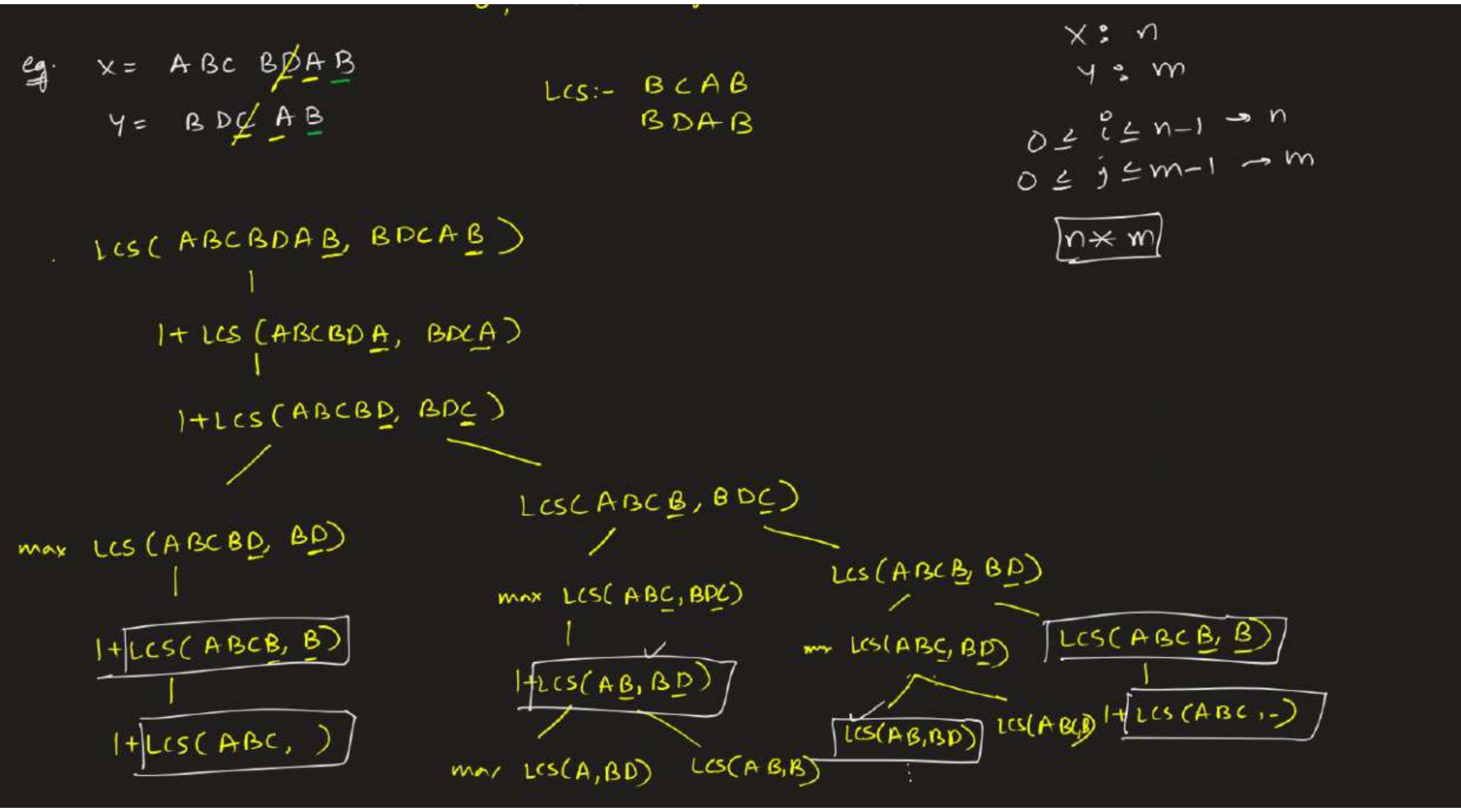
Recursive Approach to LCS:

Optimal substructure:

- If the last charcter of X and Y match, then the problem reduces in finding the LCS of X [1... n-1] and Y [1... m-1]
- If the last characters don't match, the LCS is the maximum of the LCS obtained by either excluding the last character of X or of Y.

$$LCS(X[0 \ldots i], Y[0 \ldots j]) = \begin{cases} 1 + LCS(X[0 \ldots i-1], Y[0 \ldots j-1]), & \text{if } X[i] = Y[j] \\ \underline{\max}(LCS(X[0 \ldots i-1], Y[0 \ldots j]), & \text{if } X[i] \neq Y[j] \\ \quad LCS(X[0 \ldots i], Y[0 \ldots j-1])) & \\ 0, & i<0 \text{ or } j<0 \end{cases}$$


0 1 2
A B C
0 1
B D
0 ≤ i ≤ 2
0 ≤ j ≤ 1
0 1
0 1
AB, BD
0 1 2 0
ABC, B
AB, B
A, BD
ABC, -
AB, B
B
1+ A, -
-, BD
A, B
. . .



eg. x = ABC B~~D~~AB

y = BD~~C~~ AB

LCS:- BCAB

BDAB

X : n

Y : m

$0 \le i \le n-1 \rightarrow n$

$0 \le j \le m-1 \rightarrow m$

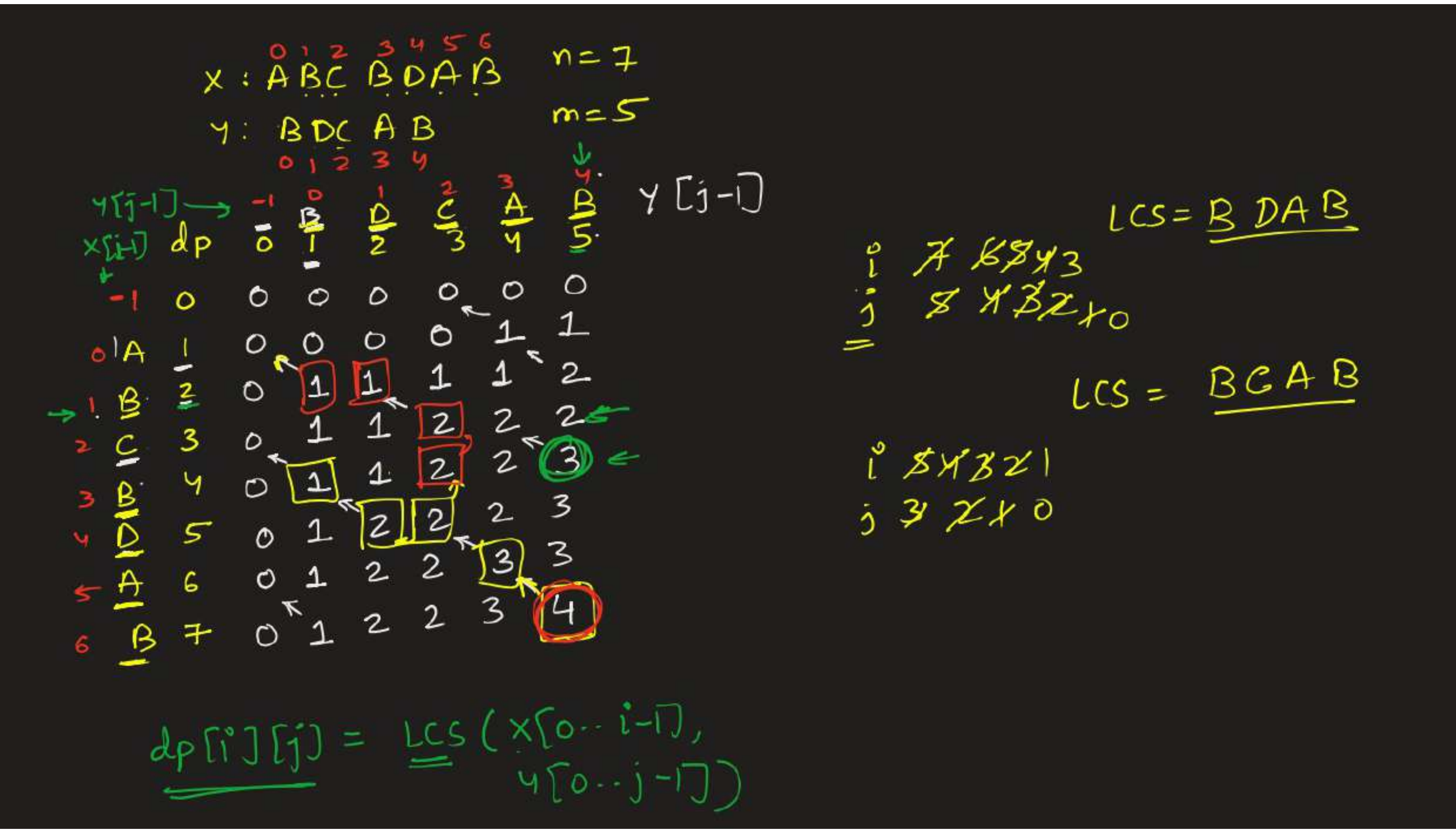
$\boxed{n * m}$

Dynamic Programming approach

```
function LCS (X, Y):
    n = length(X)
    m = length (Y)
    dp = matrix of size (m+1)x(n+1) with
         first row & first column
         initialized to 0

    for (i: 1 → n):           }  O(n* m)
        for (j: 1 → m):       }
            if X[i-1] == Y[j-1]:
                dp[i][j] = dp[i-1][j-1] + 1
            else:
                dp[i][j] = max(dp[i][j-1], dp[i-1][j]

    return dp[n][m]
```



X : ABC BDAB (indices 0 1 2 3 4 5 6) n = 7

Y : BDC AB (indices 0 1 2 3 4) m = 5

y[j-1] →		-1	0 B	1 D	2 C	3 A	4 B
x[i-1]	dp	0	1	2	3	4	5
-1	0	0	0	0	0	0	0
0 A	1	0	0	0	0	1	1
1 B	2	0	1	1	1	1	2
2 C	3	0	1	1	2	2	2
3 B	4	0	1	1	2	2	3
4 D	5	0	1	2	2	2	3
5 A	6	0	1	2	2	3	3
6 B	7	0	1	2	2	3	4

Y [j-1]

LCS = BDAB

i ~~7~~ ~~6~~ ~~5~~ ~~4~~ 3

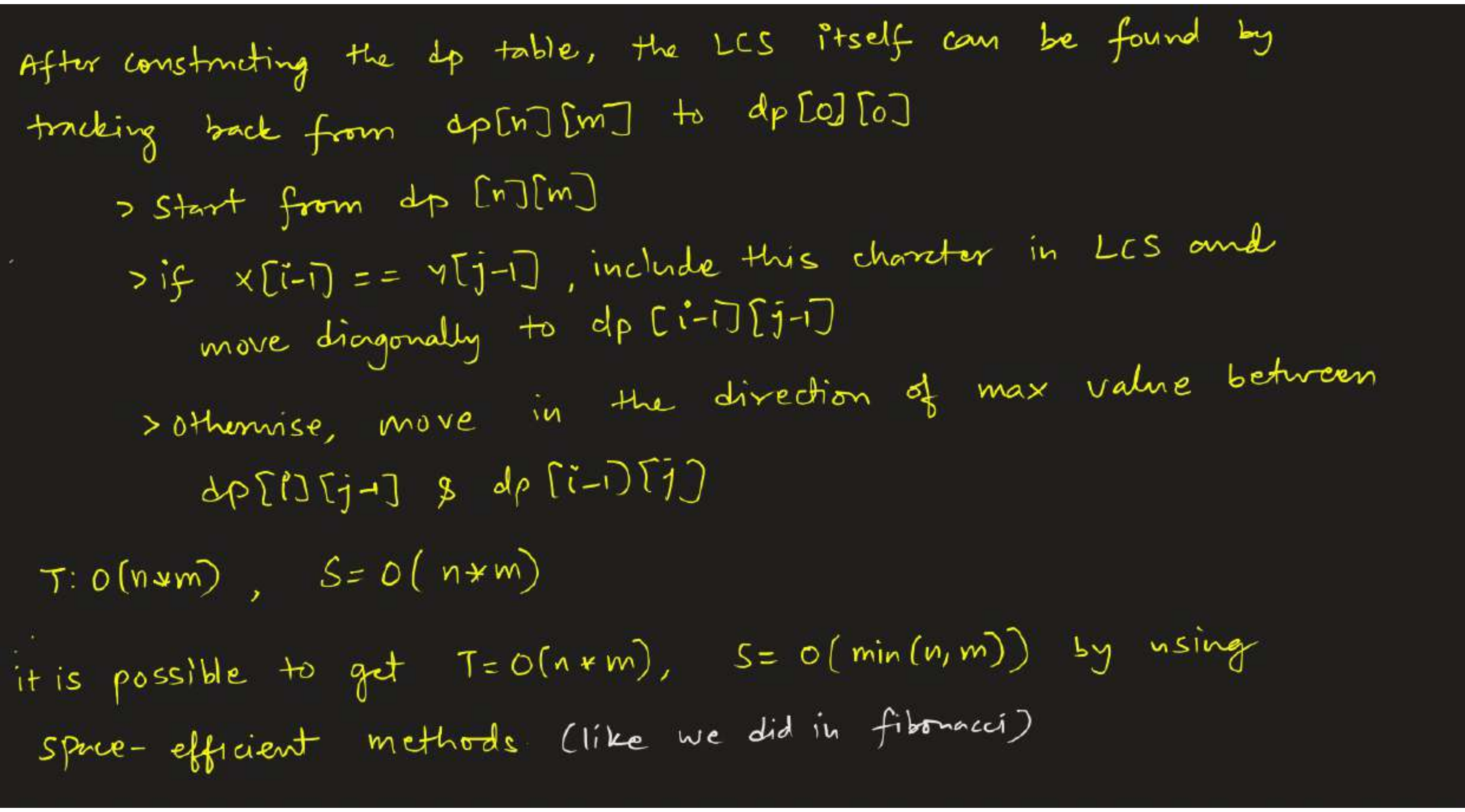
j ~~5~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0

LCS = BCAB

i ~~5~~ ~~4~~ ~~3~~ ~~2~~ 1

j ~~3~~ ~~2~~ ~~1~~ 0

dp[i][j] = LCS (X[0..i-1],
Y[0..j-1])



After constructing the dp table, the LCS itself can be found by tracking back from dp[n][m] to dp[0][0]

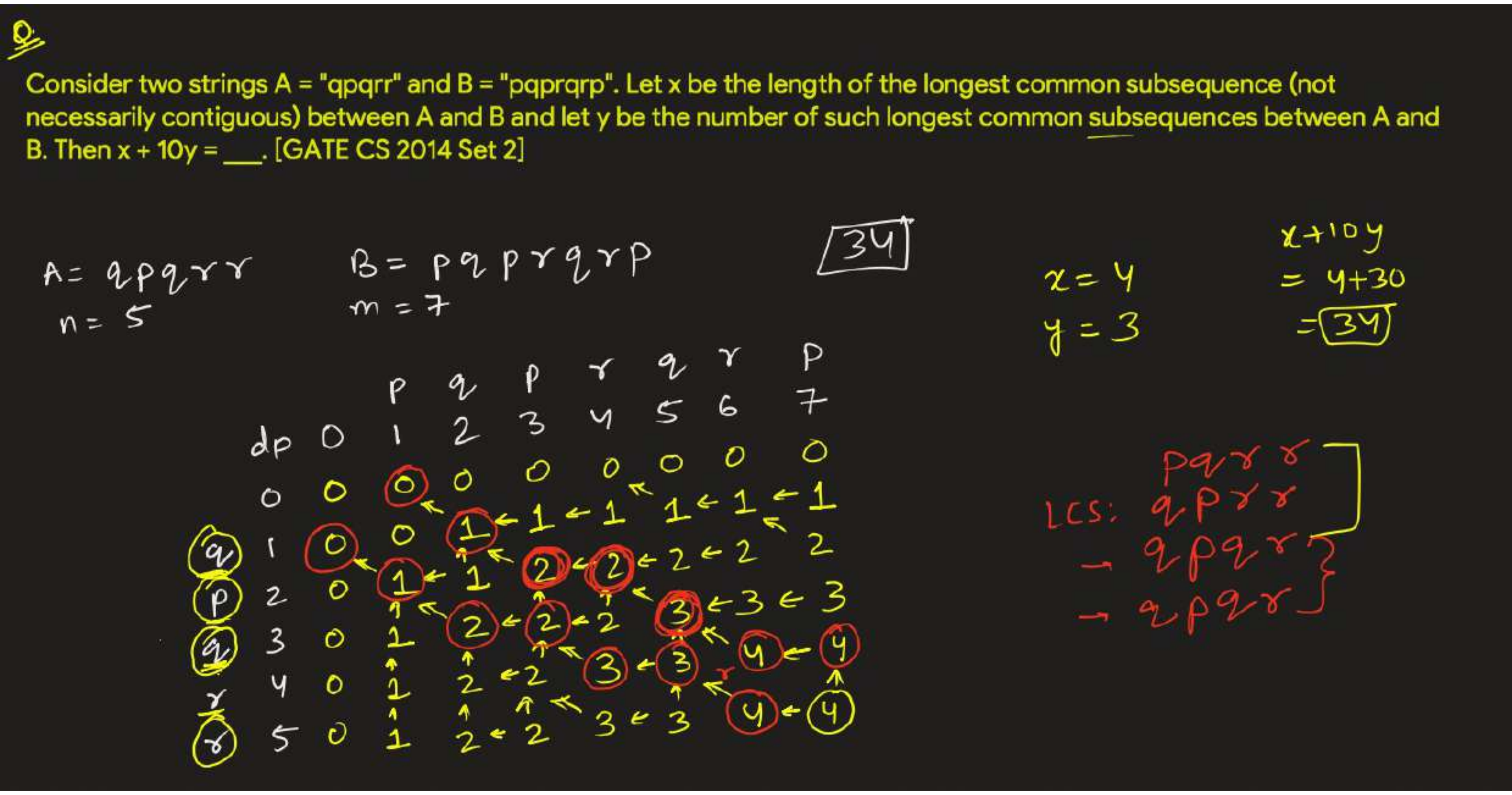
> Start from dp [n][m]

> if x[i-1] == y[j-1] , include this charcter in LCS and move diagonally to dp [i-1][j-1]

> otherwise, move in the direction of max value between dp[i][j-1] $ dp [i-1][j]

T: O(n*m) , S= O(n*m)

it is possible to get T= O(n * m), S= O(min(n, m)) by using space- efficient methods (like we did in fibonacci)



Q.

Consider two strings A = "qpqrr" and B = "pqprqrp". Let x be the length of the longest common subsequence (not necessarily contiguous) between A and B and let y be the number of such longest common subsequences between A and B. Then x + 10y = ___. [GATE CS 2014 Set 2]

A = q p q r r
n = 5

B = p q p r q r p
m = 7

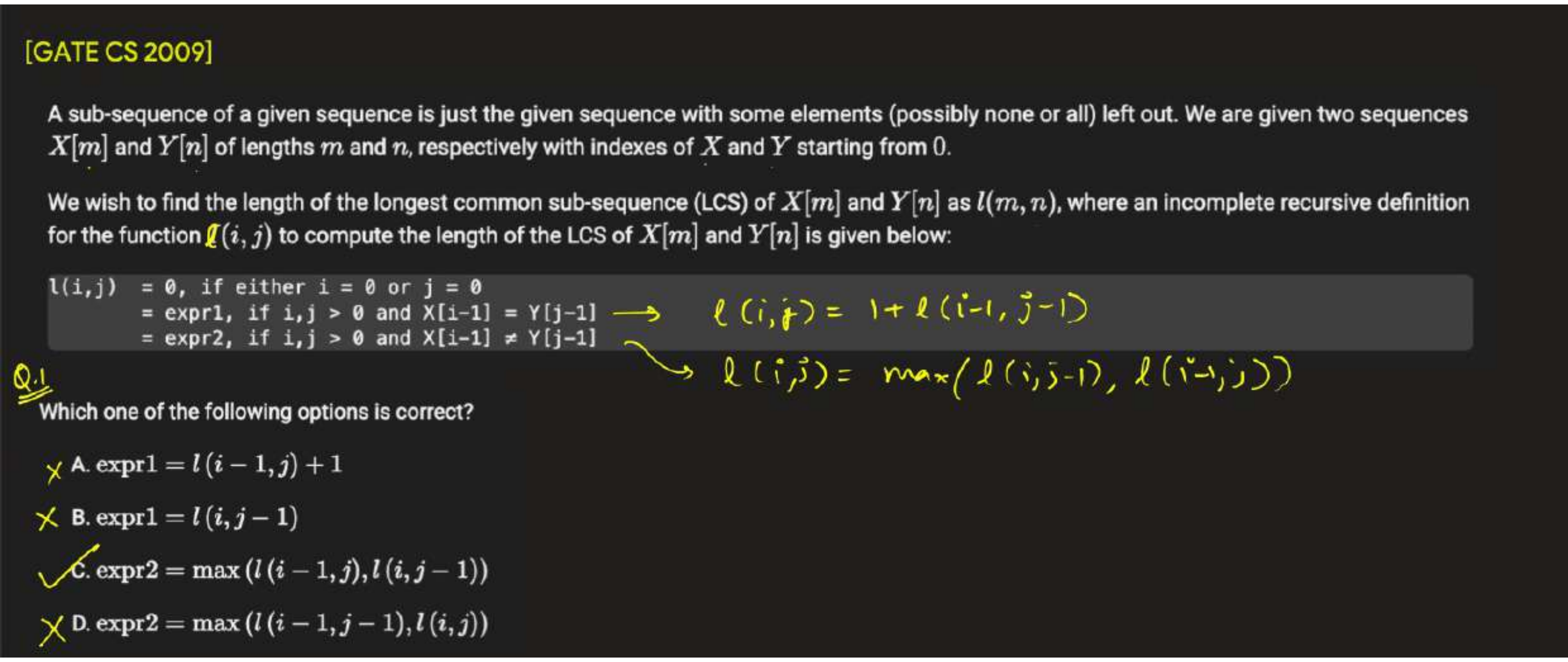
$\boxed{34}$

			p	q	p	r	q	r	p
	dp	0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
q	1	0	0	1	1	1	1	1	1
p	2	0	1	1	2	2	2	2	2
q	3	0	1	2	2	2	3	3	3
r	4	0	1	2	2	3	3	4	4
r	5	0	1	2	2	3	3	4	4

x = 4
y = 3

x + 10y
= 4 + 30
= $\boxed{34}$

LCS:
pqrr
qprr
→ qpqr
→ qpqr



[GATE CS 2009]

A sub-sequence of a given sequence is just the given sequence with some elements (possibly none or all) left out. We are given two sequences $X[m]$ and $Y[n]$ of lengths m and n, respectively with indexes of X and Y starting from 0.

We wish to find the length of the longest common sub-sequence (LCS) of $X[m]$ and $Y[n]$ as $l(m,n)$, where an incomplete recursive definition for the function $l(i,j)$ to compute the length of the LCS of $X[m]$ and $Y[n]$ is given below:

```
l(i,j)  = 0, if either i = 0 or j = 0
        = expr1, if i,j > 0 and X[i-1] = Y[j-1]
        = expr2, if i,j > 0 and X[i-1] ≠ Y[j-1]
```

→ $l(i,j) = 1 + l(i-1, j-1)$

→ $l(i,j) = \max(l(i,j-1), l(i-1,j))$

Q.1

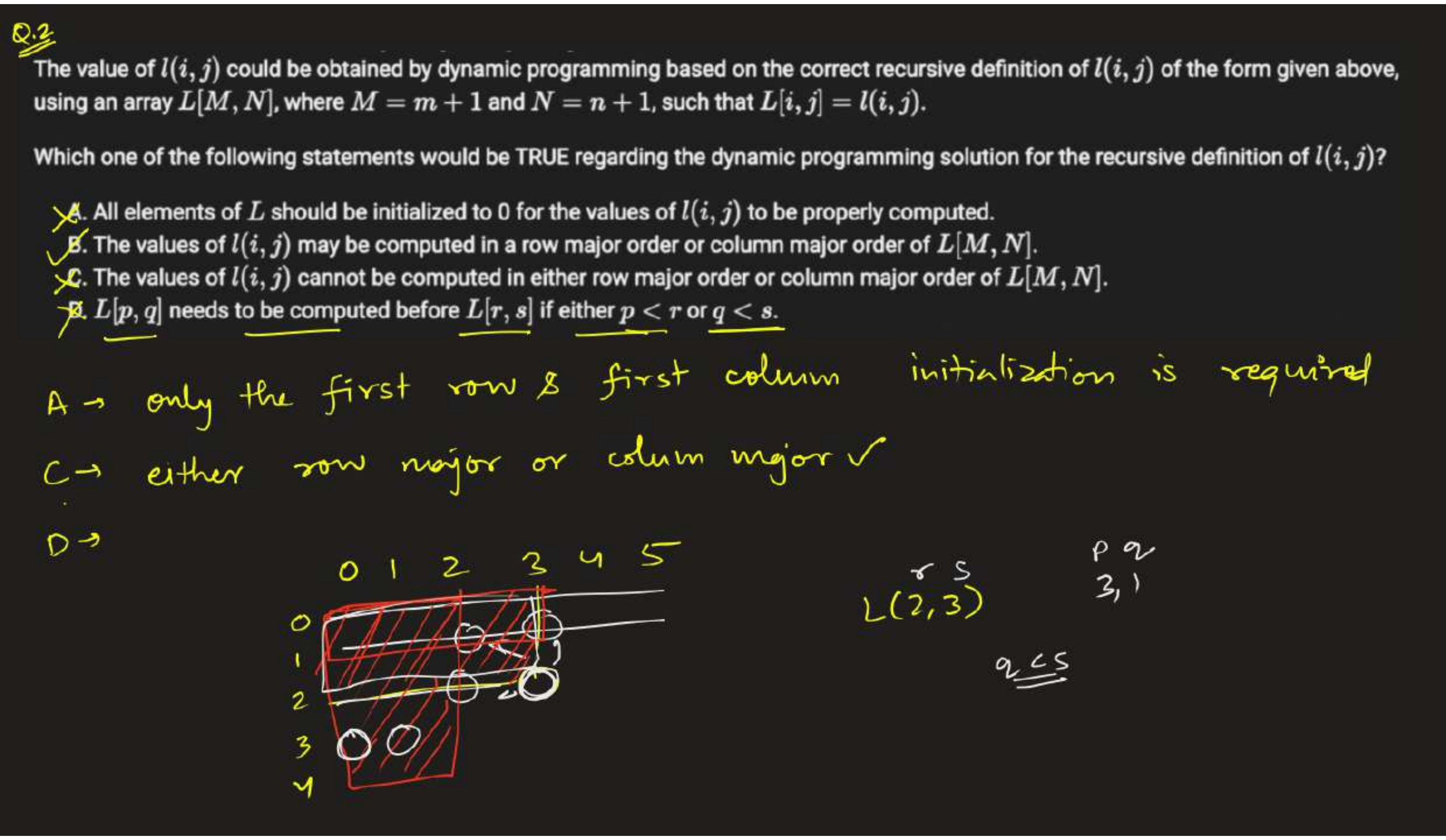
Which one of the following options is correct?

✗ A. $\text{expr1} = l(i-1, j) + 1$

✗ B. $\text{expr1} = l(i, j-1)$

✓ C. $\text{expr2} = \max(l(i-1, j), l(i, j-1))$

✗ D. $\text{expr2} = \max(l(i-1, j-1), l(i, j))$



Q.2

The value of $l(i, j)$ could be obtained by dynamic programming based on the correct recursive definition of $l(i, j)$ of the form given above, using an array $L[M, N]$, where $M = m + 1$ and $N = n + 1$, such that $L[i, j] = l(i, j)$.

Which one of the following statements would be TRUE regarding the dynamic programming solution for the recursive definition of $l(i, j)$?

- ✗ A. All elements of L should be initialized to 0 for the values of $l(i, j)$ to be properly computed.
- ✓ B. The values of $l(i, j)$ may be computed in a row major order or column major order of $L[M, N]$.
- ✗ C. The values of $l(i, j)$ cannot be computed in either row major order or column major order of $L[M, N]$.
- ✗ D. $L[p, q]$ needs to be computed before $L[r, s]$ if either $p < r$ or $q < s$.

A → only the first row & first column initialization is required

C → either row major or colum major ✓

D →

r s
L(2,3)

p q
3, 1

q < s