Recap: LCS longest common subsequence

s_1 : _____
s_2 : _____
} pick chars that are in same order in both the strings.

o/p: length of LCS.

```
int lcs(s1, s2, m, n):
DP  if m == -1 || n == -1:
        return 0
*   if memo[m][n] != -1: return memo[m][n]
    if s1[m] == s2[n]:
        memo[m][n] = 1 + lcs(s1, s2, m-1, n-1)
    else
        memo[m][n] = max{ lcs(s1, s2, m-1, n), lcs(s1, s2, m, n-1) }
    return memo[m][n]
```

int memo[m+1][n+1] = {-1}

T.C. $O(m \cdot n)$

len → if → keep track maxlen

5 variations:

① longest common substring → jump → return 0.

② SCS, shortest common supersequence s_1, s_2 → scs

EASY len s.t. s_1 and s_2 are subsequences of scs.

$$len = |m + n - lcs|$$

③ min no. of operations to convert $s_1 \rightarrow s_2$.

ops: 1 insertion, 1 deletion

$$\#min\ ops = |m + n - 2 * lcs|$$

④ longest palindromic subsequence.

↳ lcs(s, s.reverse())

⑤ min ops required to make s palindrome.

$$\#\ min\ ops = |m - lcs| \rightarrow$$ either insertions or deletions.

TOUGH. the actual LCS. | content with the length.

LCS itself !!

(6) Give me the LCS.

s_1 = abacd ⑥

s_2 = aabdccd ⑧

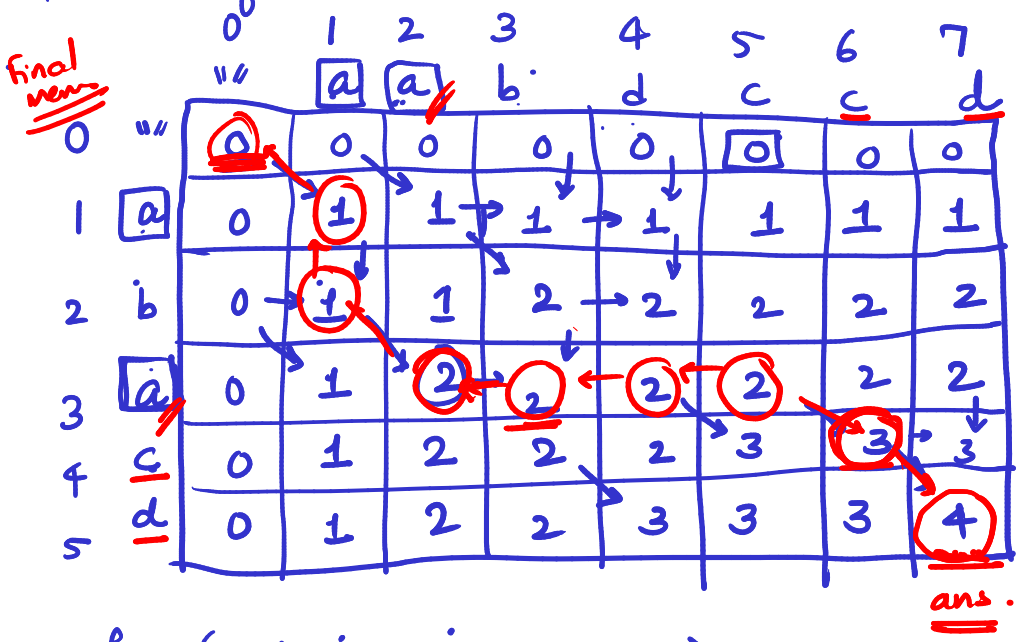
} LCS → 4

→ {a, b, c, d}

→ {a, a, c, d}

one of these !!

Take help of memo table.



a a c d *

Manual fill.

m+1
n+1

1. Base cond:

if str1 finishes || str2 finishes:

return 0

2. if $s_1[i] == s_2[j]$:

. memo[i][j] =

1 + memo[i-1][j-1]

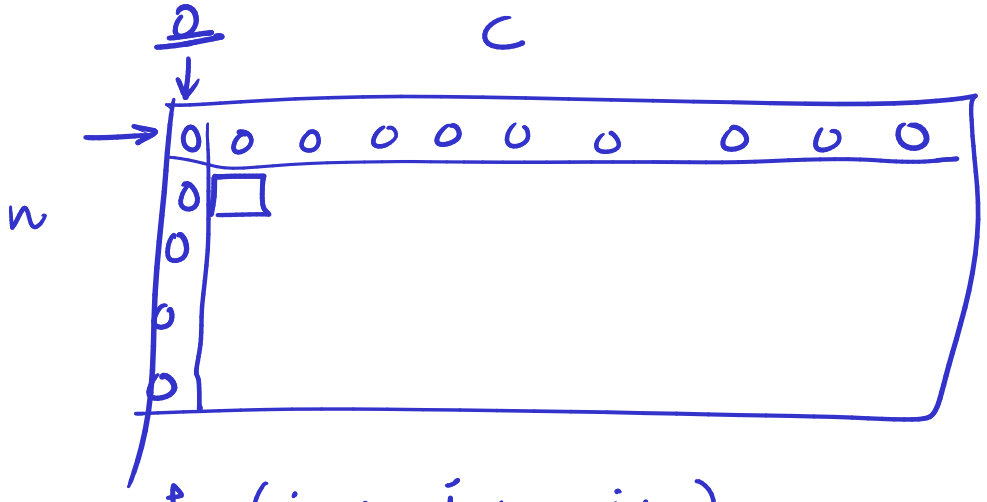
else:

memo[i][j] =

max(memo[i-1][j], memo[i][j-1])

```
for (int i=1 ,i<= m, i++)
    for (int j=1, j<= n, j++)
```

Knapsack → tabulation | No recursion



Tabulation ✗

Unless Recursion.

```
for (i=1, i≤n, i++)
    for (j=1, j≤C, j++)
        if w[i] > j :
            memo[i][j] = memo[i-1][j]
        else
            memo[i][j] = max( memo[i-1][j], p[i] +
                memo[i-1][j- w[i]] )
```

LCS | use memo.

i=m, j=n → address of final cell → end cell

```
while (i>0 || j>0):
    if s1[i] == s2[j]:
        ans += s1[i]  → char
        i-- j--
    else:                  *
        if j>0 : j--
        else   : i--
```

→ reach 1st cell
→ char. move diag.
→ otherwise i or j

str ans !

Leetcode

LCS.

↓

+

Q. —domain idea→ Recurrence Relation idea → Recursion → DP

WA TLE

normal TCs should pass

⑦ Longest repeating sequence :

LRS → 3 {a b d}.

+1 condition in if.

s_1 = a a b b c d d

```
LCS(s, s, m, n):
    if m==0 || n==0:
        return 0
    if s[m] == s[n] and m≠n:
        → 1 + lcs(s,s,m-1,n-1)
    else
        → max(lcs(s,s,m,n-1), lcs(s,s,m-1,n))
```

⑧ SCS → string itself !

⑨ This was an independent question.

Today it is a Variation !

arr = [10, 9, 2, 5, 3, 7, 101, 18]

Subsequence = [2, 3, 7, 18]

len = 4

increasing →

sorted ↑

Longest Increasing Subsequence.

LIS = LCS (arr, sort(arr))