DP Concepts & Questions

Video 21

भाषण
(Motivation)

"Always take out some time for other important topics → System Design
→ OOPs
→ Design Patterns"

"Market has started to expect these from 1-2 years exp. as well"

cswithMIK → Twitter
Facebook, Instagram → codestorywithMIK
whatsapp → codestorywithMIK

Done
- 1-D based DP
- 2-D based DP

Progress
- String based DP
- Grid based DP
- Game Strategy

We'll do :-
(·) RECURSION + MEMOIZATION (Top Down)
(·) Bottom UP.
(·) Time & Space

DP on Strings :-

- Longest Common Subsequence (LCS)
- Print LCS
- → Edit Distance
- Shortest common Supersequence. (SCS)
- Print SCS
- → Longest Palindromic Subsequence
- → More comings..... :)

(Hard Qns → Easy) many more...

Print Shortest Common Supersequence ...

1092. Shortest Common Supersequence

Hard | Topics | Companies | Hint

Given two strings `str1` and `str2`, return *the shortest string that has both* `str1` *and* `str2` *as* ***subsequences***. If there are multiple valid strings, return **any** of them.

A string `s` is a **subsequence** of string `t` if deleting some number of characters from `t` (possibly `0`) results in the string `s`.

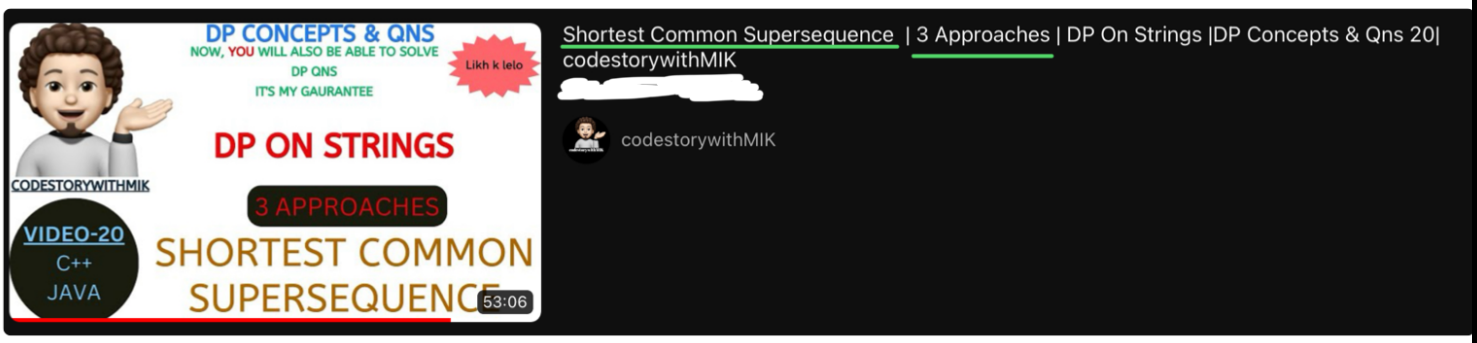
Example :- S1 = "abac"

S2 = "cab"

Output = "cabac"

SCS

(•) You can also watch "Print Longest Common Subsequence"

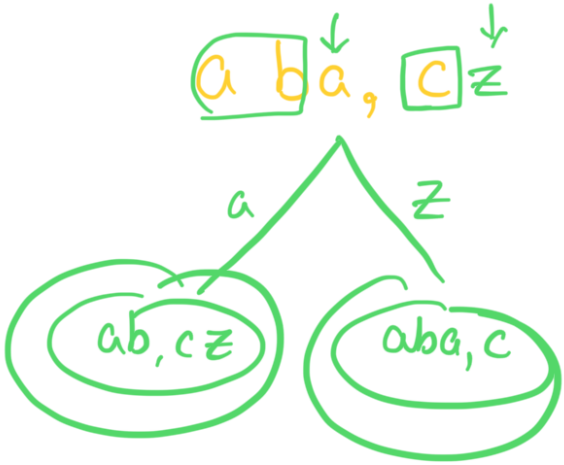


Bottom UP

```
class Solution {
    public:
    //Function to find length of shortest common supersequence of two strings.
    int shortestCommonSupersequence(string s1, string s2, int m, int n) {
        vector<vector<int>> t(m+1, vector<int>(n+1));

        for(int i = 0; i < m+1; i++) {
            for(int j = 0; j < n+1; j++) {
                if(i == 0 || j == 0) {
                    t[i][j] = i+j;
                } else if(s1[i-1] == s2[j-1]) {
                    t[i][j] = 1 + t[i-1][j-1];
                } else {
                    t[i][j] = 1 + min(t[i-1][j], t[i][j-1]);
                }
            }
        }

        return t[m][n];
    }
};
```



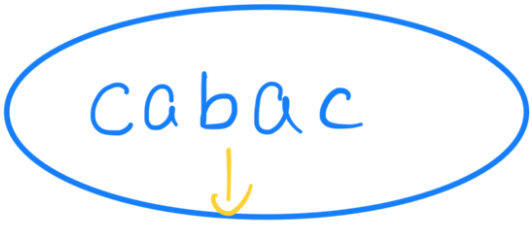
State : t[i][j] = SCS of S1 of length i & S2 of length j.

m ← S1 = abac
n ← S2 = cab

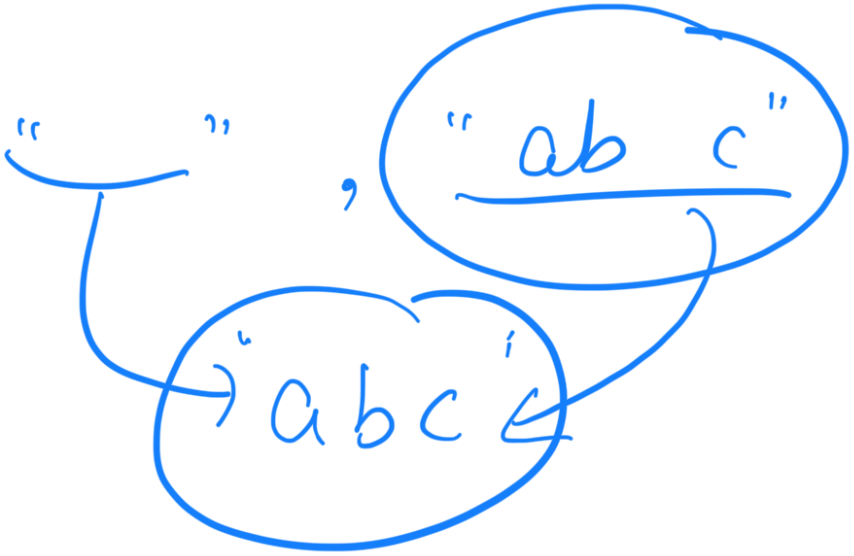
		""	c	ca	cab
		0	1	2	3
""	0	0	1	2	3
a	1	1	2	2	3
ab	2	2	3	3	3
aba	3	3	4	4	4
abac	4	4	4	5	5

i = m j = n

t[i][j]



cabac



abc ∈ "" "abc" ⟵ [0] [3]

abc ≠ "abc" "" ⟵ [3] [0]

```
while (i > 0 && j > 0) {
    if (s1[i-1] == s2[j-1]) {
        result.push_back(s1[i-1]);
        i--; j--;
    } else {
        if (dp[i-1][j] < dp[i][j-1]) {
            res.push_back(s1[i-1]);
            i--;
```

```
} else {
        result.push_back(s2[j-1]);
        j--;
    }
}

while (i > 0) {
    result.push_back(s1[i-1]);
    i--;
}
while (j > 0) {
    result.push_back(s2[j-1]);
    j--;
}

reverse(result.begin(), result.end());

return result;
}
```