DP Concepts & Questions

Video 25

भाषण
(Motivation)

"Your education/hardwork is a rehearsal for a life that you are going to lead in future...

Do it carefully"

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

⇒ Palindrome related DP problems :-

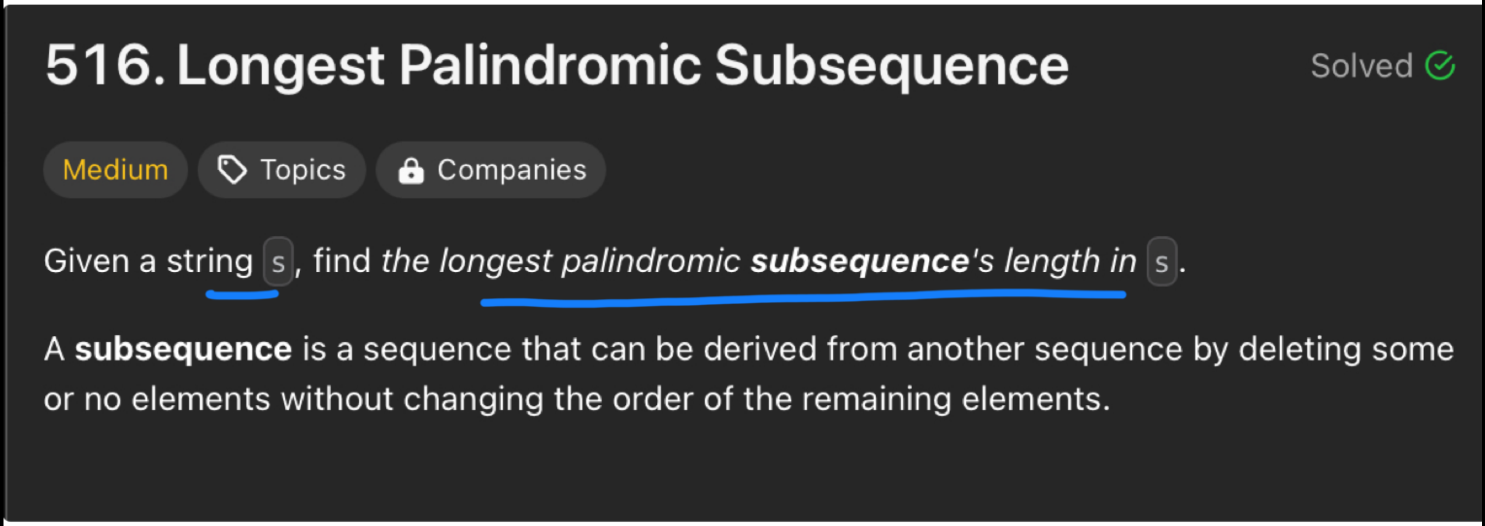
↳ Palindromic Substrings + Blueprint

↳ Longest Palindromic Substring

Longest Pallindromic Subsequence

(•) Recursion + Memoization ✓

(•) Bottom UP → Blue Print ✓



516. Longest Palindromic Subsequence

Solved ✓

Medium Topics Companies

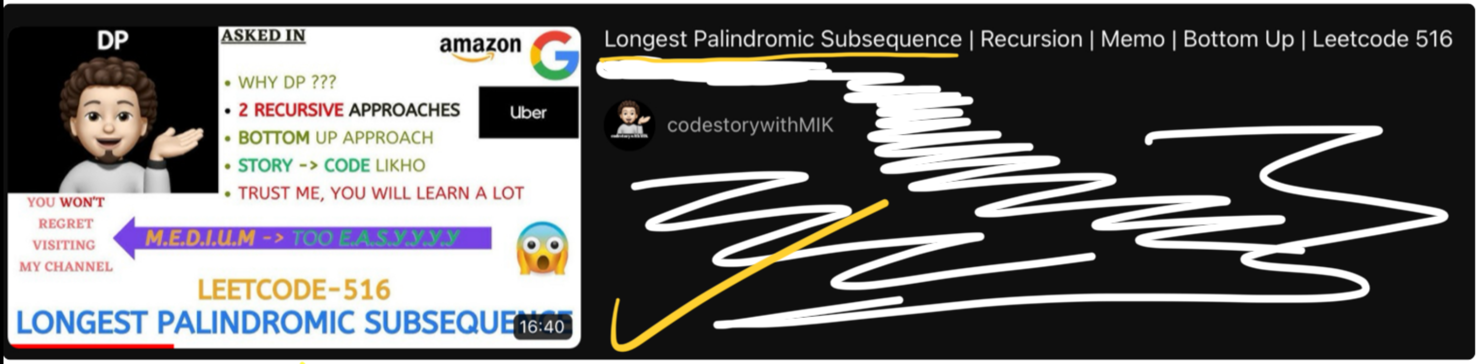
Given a string `s`, find *the longest palindromic* ***subsequence****'s length in* `s`.

A **subsequence** is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements.

Example :- s = "bbbab"

Output = 4 → "bbbb"

LINK in the Description !!!


Longest Palindromic Subsequence | Recursion | Memo | Bottom Up | Leetcode 516

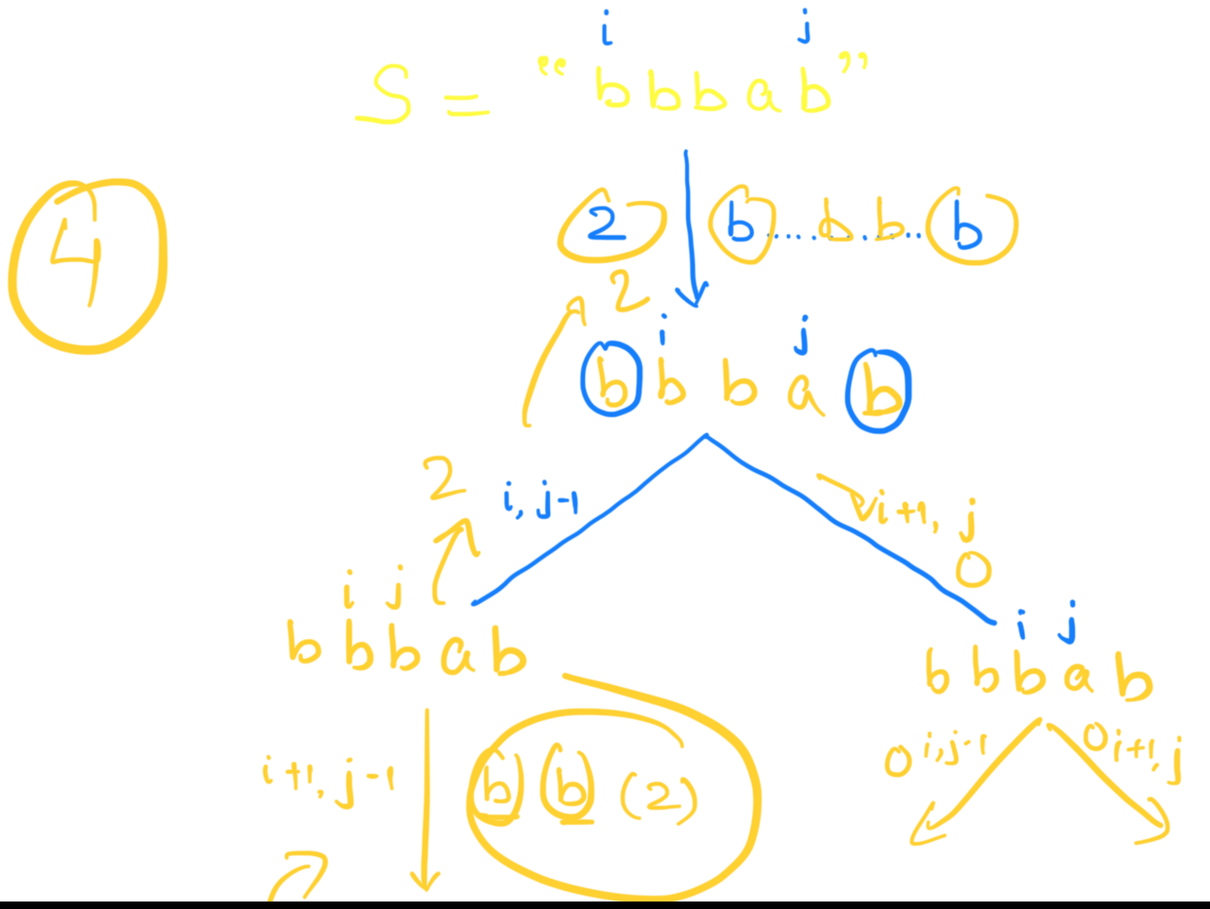
codestorywithMIK

DP
ASKED IN
amazon
Uber
• WHY DP ???
• 2 RECURSIVE APPROACHES
• BOTTOM UP APPROACH
• STORY -> CODE LIKHO
• TRUST ME, YOU WILL LEARN A LOT
YOU WON'T REGRET VISITING MY CHANNEL
M.E.D.I.U.M -> TOO E.A.S.Y.Y.Y.Y
LEETCODE-516
LONGEST PALINDROMIC SUBSEQUENCE
16:40

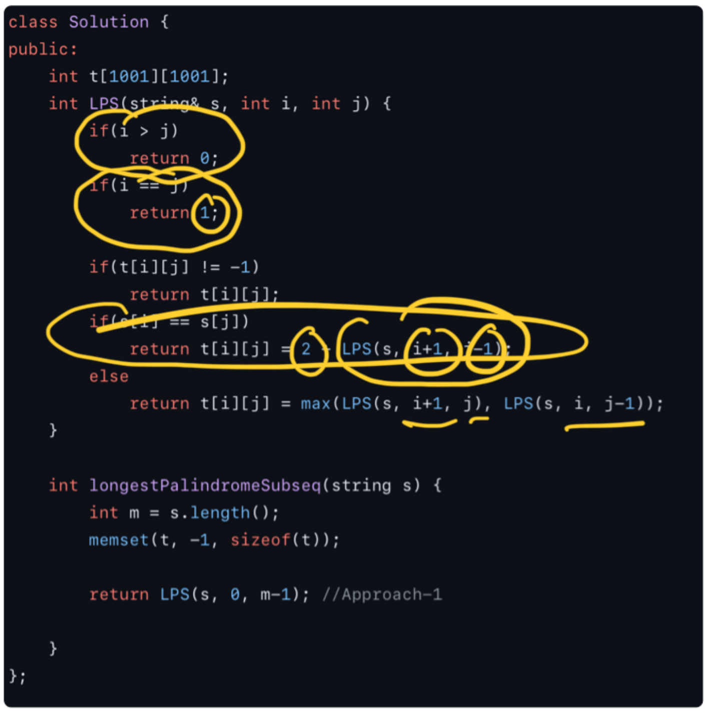
↳ Recursion + Memo → 2 ways

↳ Bottom up

We will solve it using the same Palindrome BLUEPRINT



b $\overset{j}{b}$ $\overset{i}{b}$ a b



```cpp
class Solution {
public:
    int t[1001][1001];
    int LPS(string& s, int i, int j) {
        if(i > j)
            return 0;
        if(i == j)
            return 1;

        if(t[i][j] != -1)
            return t[i][j];
        if(s[i] == s[j])
            return t[i][j] = 2 + LPS(s, i+1, j-1);
        else
            return t[i][j] = max(LPS(s, i+1, j), LPS(s, i, j-1));
    }

    int longestPalindromeSubseq(string s) {
        int m = s.length();
        memset(t, -1, sizeof(t));

        return LPS(s, 0, m-1); //Approach-1

    }
};
```

Blueprint: (Bottom_up)

$S = "\overset{0}{b}\ \overset{1}{b}\ \overset{2}{b}\ \overset{3}{a}\ \overset{4}{b}"$ LPS

$t[i][j]$ = LPS of $S[i \dots j]$

return $t[0][n-1]$; //LPS of whole string.

		0 b	1 b	2 b	3 a	4 b
0	b	1				
1	b		1			
2	b			1		
3	a				1	
4	b					1

0, 2

$t[0][2] = x$

0, 3

t   t[0][3]=y.

•) L = 1 → LPS = 1 → Palindrome.

```
for(int L=2; L<=n ; L++) {
    for(int i=0; i<n-L+1; i++) {
        int j = i+L-1;
        if (s[i] == s[j])
            t[i][j]=2+ t[i+1][j-1];
        else {
            t[i][j]= max (t[i][j-1], t[i+1][j]);
        }
    }
}

return t[0][n-1] ;

}
```

$T.C = O(n^2)$

$S.C = O(n^2)$.