DP Concepts & Questions

Video 24

भाषण
(Motivation)

> "Don't be afraid to give up the Good to go for the Great. Trust in your journey and Keep pushing forward..."

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

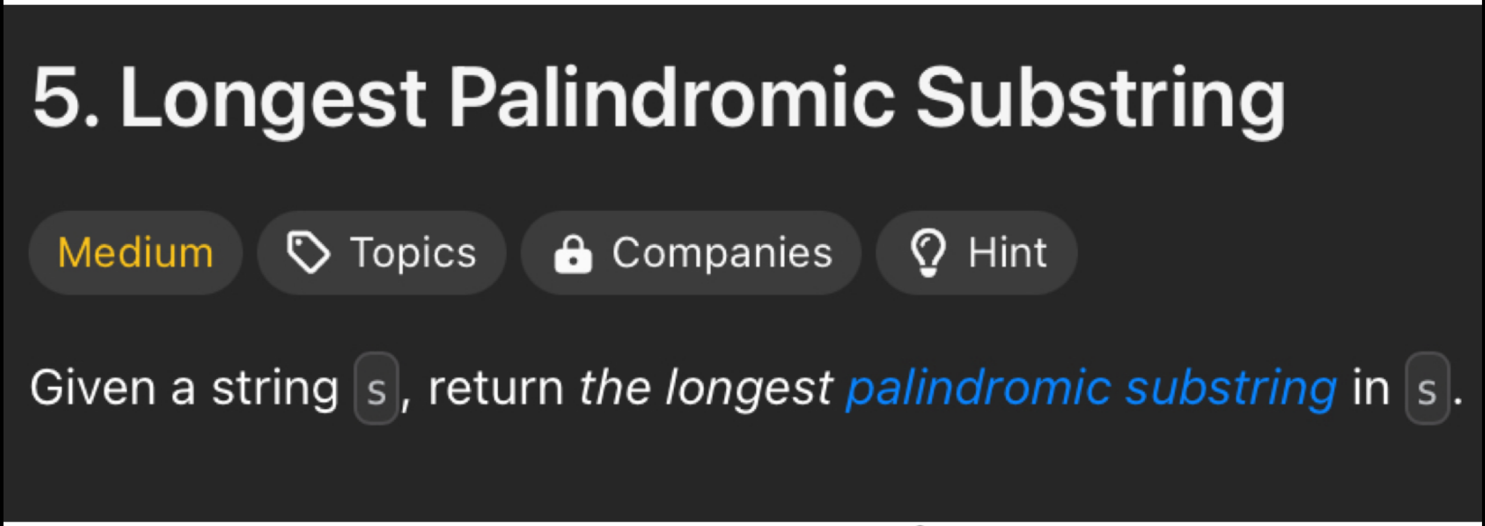
⇒ Pallindrome related DP problems :-

- └→ Palindromic Substrings + Blueprint
- └→ Longest Palindromic Substring

→ Longest Pallindromic Subsequence

(•) Recursion + Memoization ✓

(•) Bottom UP → Blue Print ✓



5. Longest Palindromic Substring

Given a string `s`, return *the longest* palindromic substring in `s`.

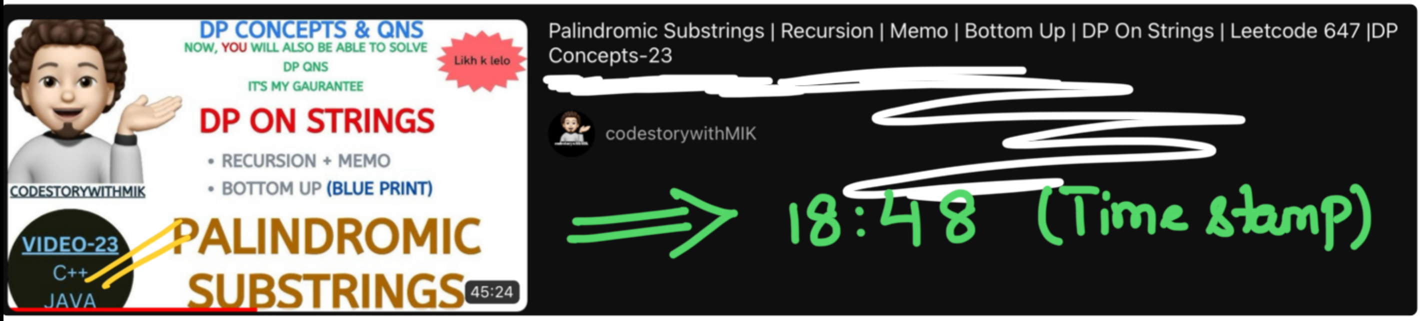
Example:- "babad"

Output:- "bab"

Bottom UP

Using same Blue Print

First go and see the intuition & thought Process behind this Blue Print.



Palindromic Substrings | Recursion | Memo | Bottom Up | DP On Strings | Leetcode 647 |DP Concepts-23

[redacted]

codestorywithMIK

⟹ 18:48 (Time stamp)

State definition:- t[i][j]

t[i][j]
- → True → S[i...j] → Palindrome
- → False → S[i..j] → Not a palindrome

✳ length = 1 का कोई substring always = Pal.

$\boxed{t[0][0] = \text{True}}$

t[1][1] = True

t[2][2] = True

t[3][3] = Tr

t[4][4] = Tr.

0 1 2 3 4

" b a b a d "

"b" → S[0][0] → f[0][0] = True.

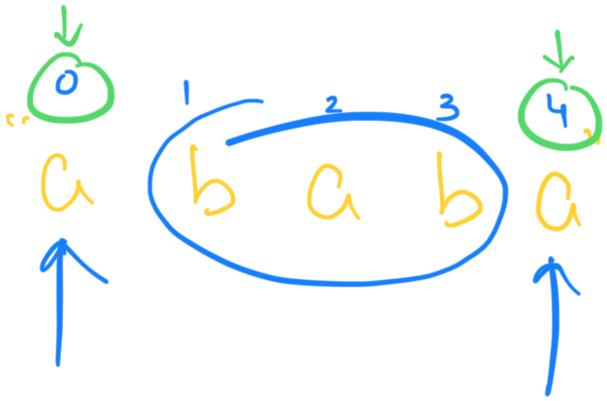
		b	a	b	a	d
		0	1	2	3	4
b	0	T	-	-	-	-
a	1	-	T	-		- .
b	2	-	.	T .	-	-
a	3	.	.	-	T	.
d	4	.	-		.	T

f

f[0][0] = True

s[0,3]

f[0][3] = F



s(i) == s[j]

f[i+1][j-1] = Tr

if (s[i] == S[j] && f[i+1][j-1] == True) f[i][j]= True;

L = 1 → True.

L >= 2

```
  0 1 2 3 4 5 6
" a b a c a b a "
```

```
maxL = 1 ; idx = -1;

for(L= 2; L<=n ; L++) {
    for (i=0 ; i<n-L+1 ; i++) {

        j= i+L-1;

        if(S[i]==S[j] &&L==2){
            f[i][j] = True; maxL=2
                            idx=i;
        }else if(S[i]==S[j] && f[i+1][j-1])
            f[i][j] = True;
            if(j-i+1 > maxL) {
            } maxL= j-i+1; idx=i;
        } else {
            f[i][j]= False;
        }
}
```

Blueprint.

```
}
```

maxL, idx

```
return s.substr(idx, maxL);
```