DP Concepts & Questions

Video 28

भाषण
(Motivation)

" CONSISTENCY is the Key to mastery.
Show up everyday, even when it's tough... "

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

Palindrome Partitioning - I

Palindrome Partitioning - II

(•) Recursion + Memoization

(•) Bottom UP → Blue Print

132. Palindrome Partitioning II

Hard | Topics | Companies

Given a string s, partition s such that every substring of the partition is a palindrome.

Return *the* ***minimum*** *cuts needed for a palindrome partitioning of* s.

Examples :- S = "aab"

a a | b

Output = 1

s = "aba"

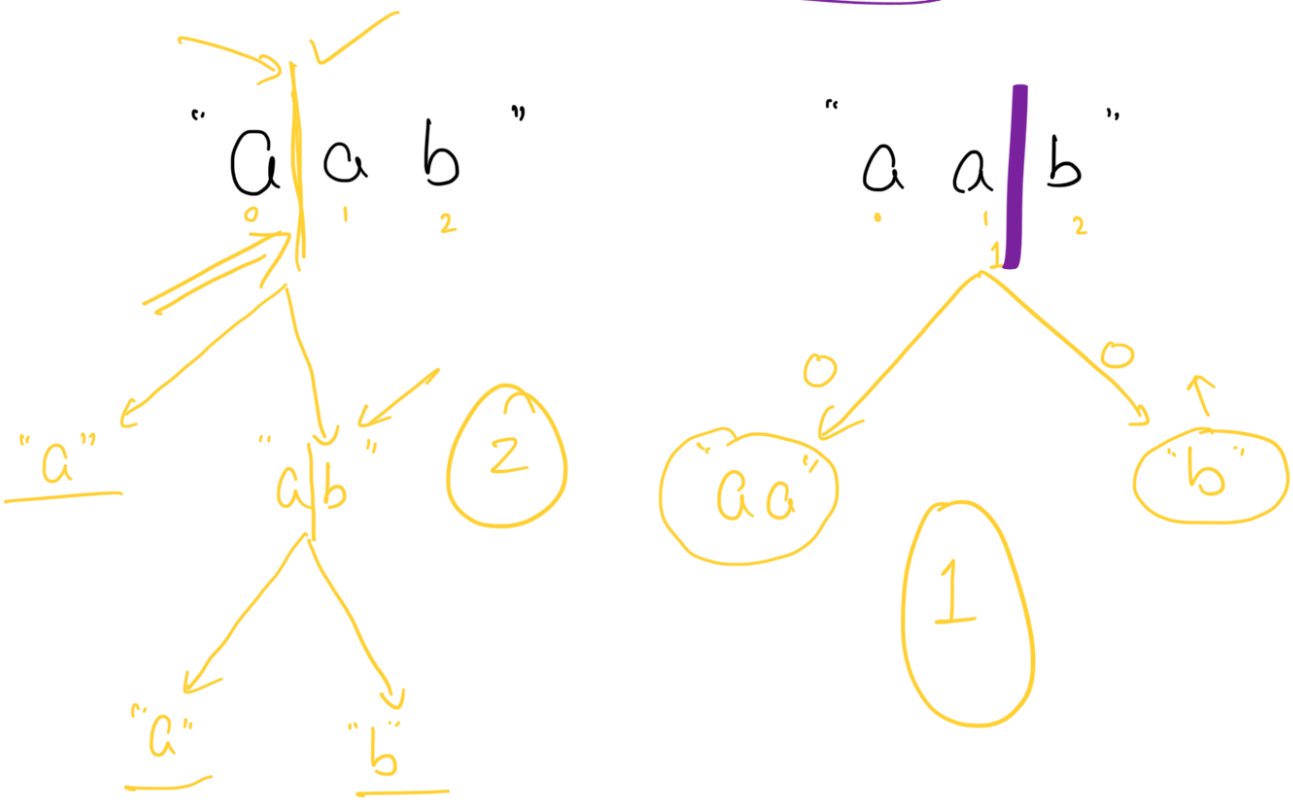
"aba"

Output = 0

s = "abcb" → "a" | "bcb"

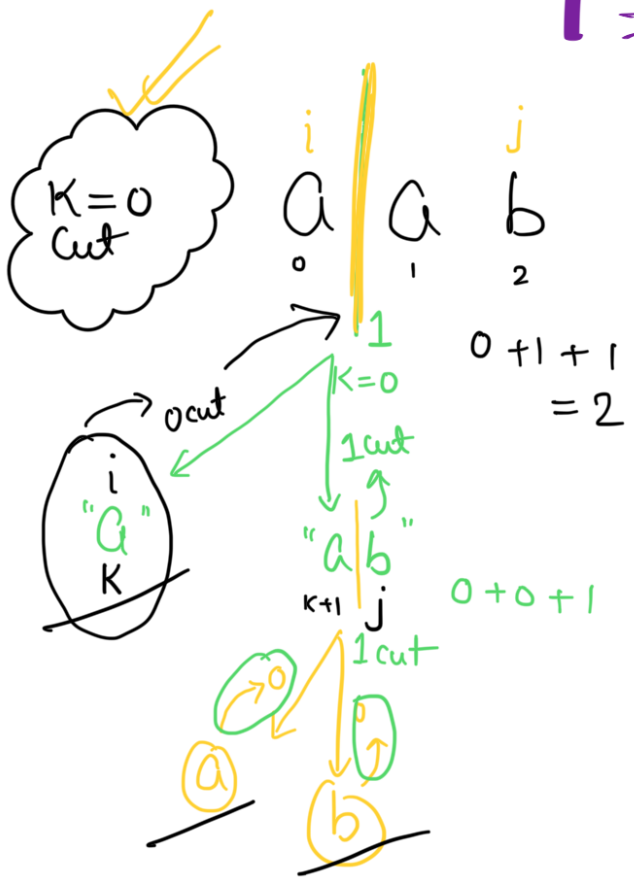
Output = 1

Recursion + Memoization



options → 2

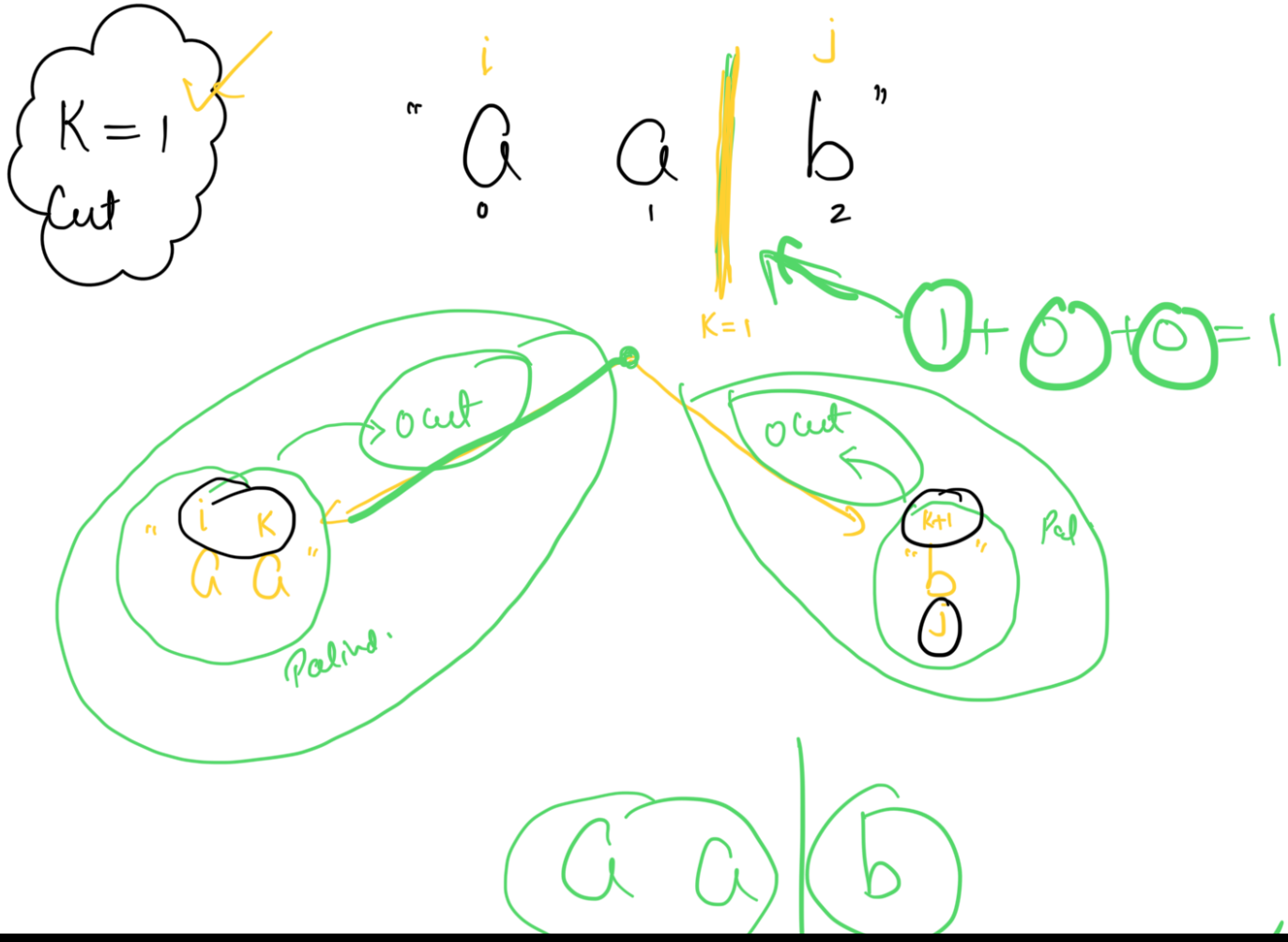
options ⟹ Recursion



0+1+1
=2

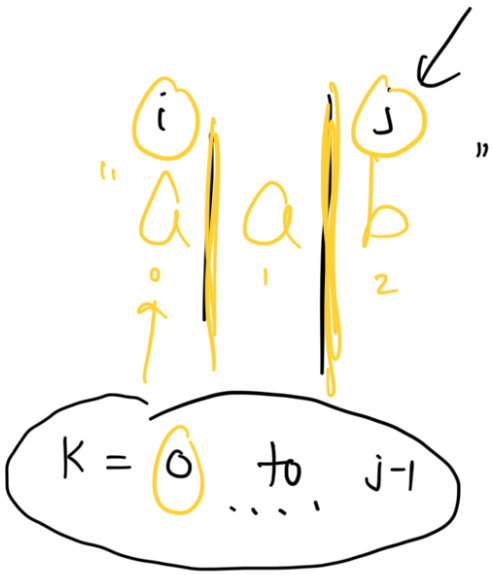
a | a | b

Cuts=2



cut = 1

K = 2



K = 0

K = 1

$\boxed{\text{Solve}(s, i, j);}$

```
int Solve(s, i, j) {

    if(isPalind(s, i, j)) {
        return 0; //No cut req.
    }
    int result = INT_MAX;
```

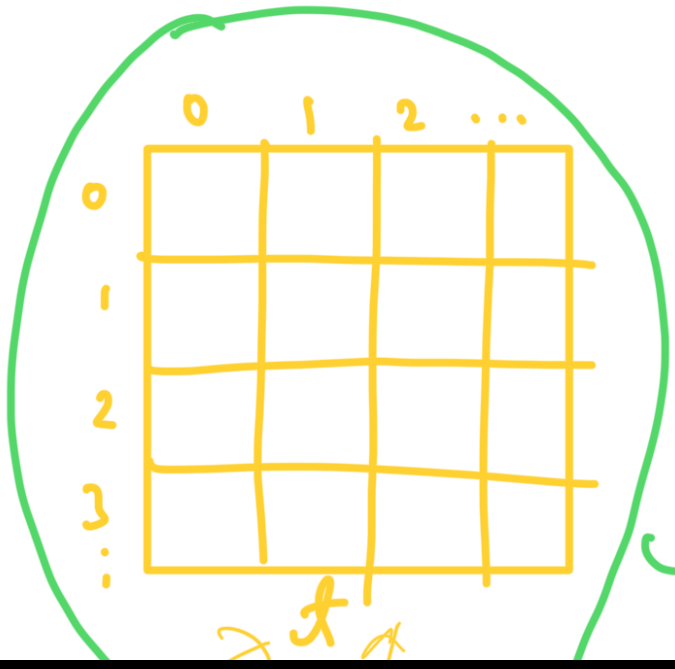
```
for (int k = i ; k <= j-1 ; k++) {
        int temp = 1 + Solve(s, i, k) +
                       Solve(s, k+1, j);
        result = min(result, temp);
    }

    return result;
}
```

$t[i][j]$

Bottom UP

(BluePrint)

(Palindrome DP Problem)



$t[i][j] = True$

$S[i \ldots j] \rightarrow$ Palindrome

0	1	2	3
a	b	c	b

$n = 4$

i

$1 + dp[2]$

$S[0 \text{ to } i] = \text{"ab"}$

if ($t[k+1][i] == \text{True}$
&& $1 + dp[k] < dp[i]$)
$\boxed{dp[i]} = 1 + dp[k]$;

$0 \to 0 = 0$ cuts

$0 \to 1 =$

$dp[2] =$

	0	1	2	3
dp	0	1	2	1

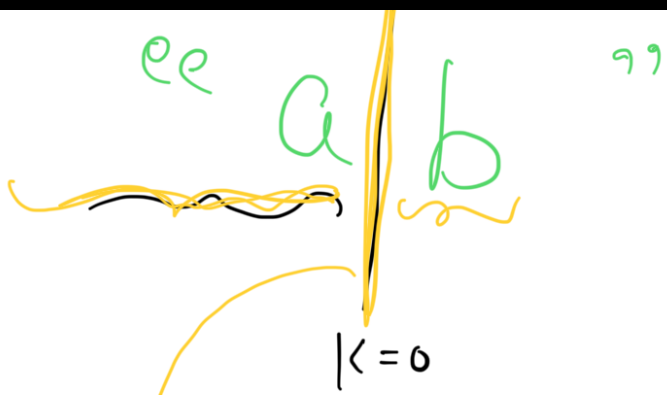
$dp[3] = 1$

$dp[i]$ = min cuts to split a $S[0 \ldots i]$ into pal.

return $dp[3]$ i.e $dp[n-1]$;

$S[0 \ldots n-1]$

0 | i



$f[K+1][i] = T$

$dp[K]$

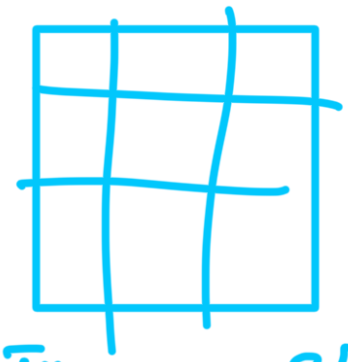
$dp[i] = \min(dp[i], 1 + dp[K])$;

$1 + dp[0]$

$1 + 0 = 1$

Story Points :-

① Blue Print → f



$f[i][j]$ → True → $S[i \ldots j]$ is Pal.

↳ False → S[i...j] is Not pal.

② Build dP array by trying all possible cuts at diff. indices.

dP[i] = S[0...i] → How many cuts (min) will be req.

S[0... n-1]

return dP[n-1];