DP Concepts & Questions

Video 27

भाषण
(Motivation)

"If you want to see a change, start working for it..."

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

(•) Recursion + Memoization

(•) Bottom UP → Blue Print

131. Palindrome Partitioning

Solve

Medium | Topics | Companies

"abacdd"

{"aba", "c", "dd"}

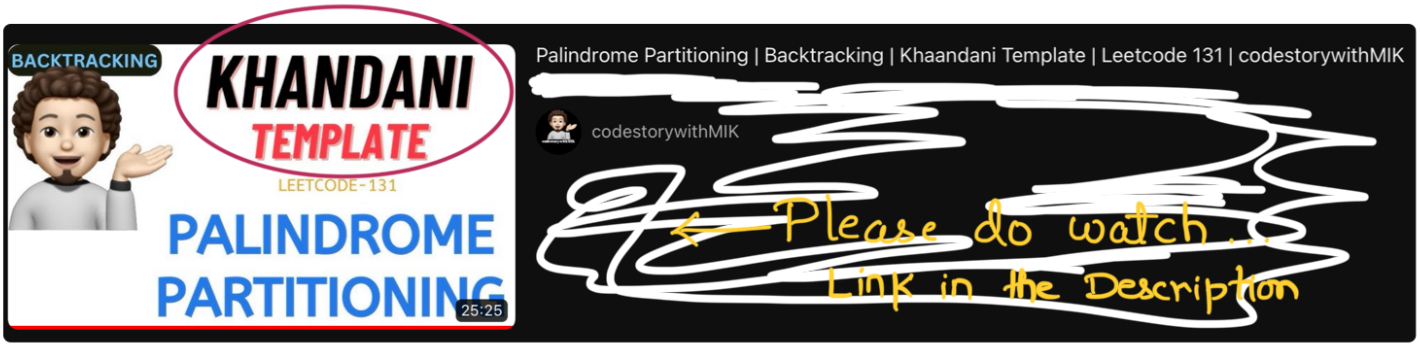
Given a string s, partition s such that every substring of the partition is a **palindrome**. Return *all possible palindrome partitioning of* s.

Backtracking.

Example :- S = "aab"

Output :- {{"a", "a", "b"}, {"aa", "b"}}

"All possible" → BACKTRACKING...



$s =$ "a a b" (0, 1, 2)

$n = 3$

$t[i][j] = T_{rue}$; $S[i..j]$ → Pall.

False $S[i...j]$ .... → Not a pa.

$t[0][1] \rightarrow$ Y.

L=1 (•) $\boxed{t[i][i] = \text{True}}$ // Diagonal Ele. "Substrin of 1 Char"

	0	1	2
0	T		
1		T	
2			T

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

"a a" (i, j)

Blue Print.

t =

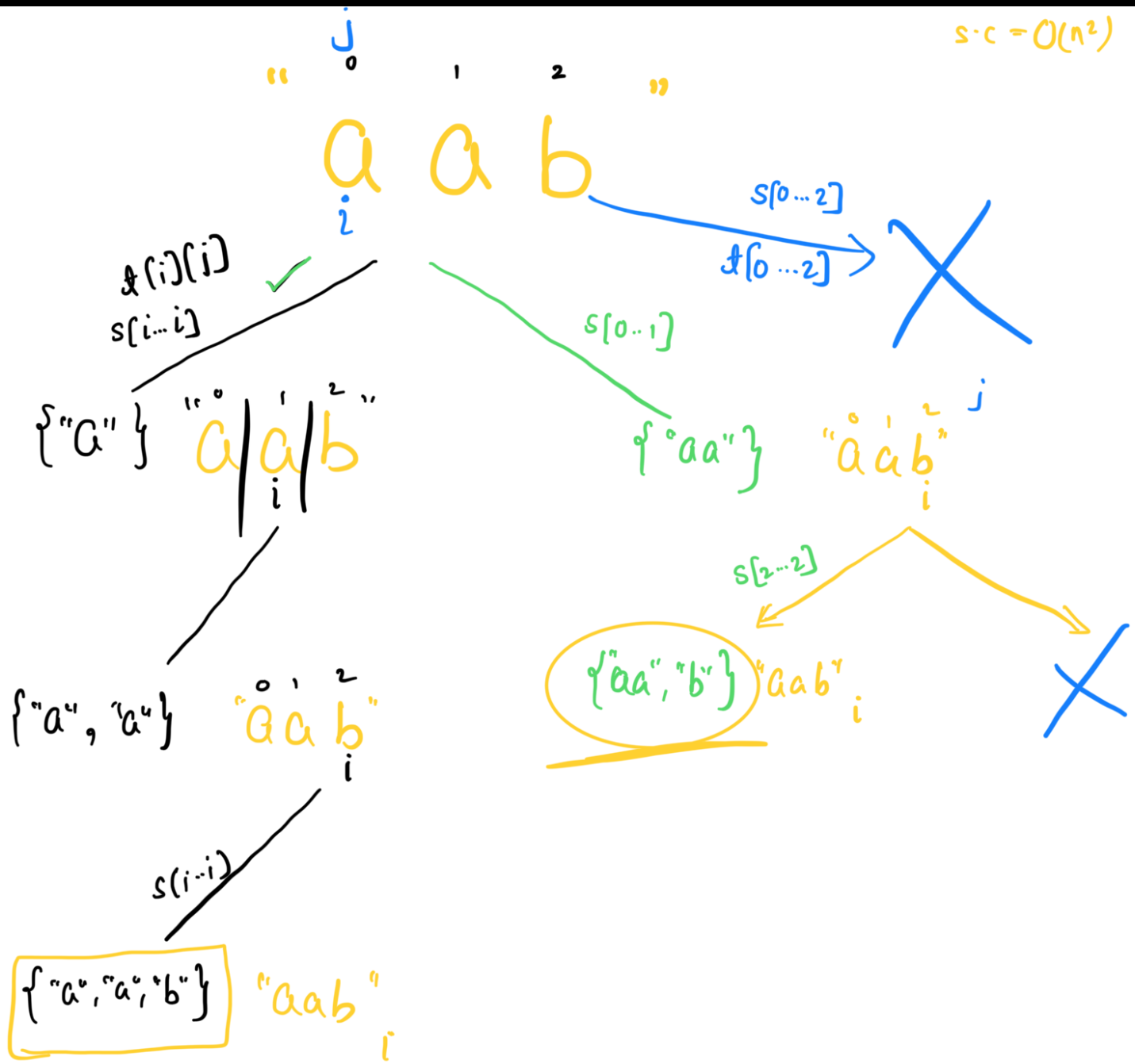
	0	1	2
0	T	T	F
1	F	T	F
2	F	F	T

3x3

S = "aab" (0, 1, 2)
n = 3

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

s.c = $O(n^2)$



result = { {"a", "a", "b"}, {"aa", "b"} }

Story to Code:-

(i) Blue print → t[i][j]

```
Solve(S, 0);

Solve(string S, int i, Part){
    if (i == s.length()) {
        result.push_back(Part);
        ret;
    }

    for (j = i; j < s.length; j++){
        if (t[i][j] == True) {//O(1)
            Part.push_back(s.substr(i, j-i+1));
            Solve(S, j+1, Partion); //ans:-
```

Con:-

Exp:

undo

path.pop_back();

}

}

}