DP Concepts & Questions

Video 14

भाषण
(Motivation)

- When you don't feel like studying, always remember

"Someone is working harder than you to get that job"

cswithMIK → Twitter

Facebook, Instagram → codestorywithMIK

- 1-D based DP

We'll do :-

(•) RECURSION

- 2-D based DP
- String based DP
- Grid based DP
- Game Strategy

MEMOIZATION (Top Down)

(•) Bottom Up

(•) Time & Space

1-D Based D.P.

↳ Fibonacci Number

↳ Climbing Stairs (N-staires Problem)

↳ House Robber-I, II (Stickler thief)

↳ Maximum Alternative Subsequence Sum.

↳ LIS and all variants

(•) LIS (longest Increasing Subsequence.

Recursion + Memo / Bottom up

video - 11 & 12

Variants

1. Maximum Length Of chain Pair . (video-13)
2. Longest String chain (POTD).

"Longest String Chain"

Leetcode - 1048

~~Medium~~ Easy

Company Tags :- Google

1048. Longest String Chain

Medium 6358 228 Add to List Share

You are given an array of `words` where each word consists of lowercase English letters.

$word_A$ is a **predecessor** of $word_B$ if and only if we can insert **exactly one** letter anywhere in $word_A$ **without changing the order of the other characters** to make it equal to $word_B$.

- For example, `"abc"` is a **predecessor** of `"abac"`, while `"cba"` is not a **predecessor** of `"bcad"`.

A **word chain** is a sequence of words `[word1, word2, ..., wordk]` with `k >= 1`, where $word_1$ is a **predecessor** of $word_2$, $word_2$ is a **predecessor** of $word_3$, and so on. A single word is trivially a **word chain** with `k == 1`.

Return *the **length** of the **longest possible word chain** with words chosen from the given list of* `words`.

Example:-

words = ["a", "b", "ba", "bca", "bda", "bdca"]

output = 4

words = ["abcd", "abxcd", "a", "ab", "abc"]

→ ["a", "ab", "abc", "abcd", "abxcd"]

output = 5

(·) (i) length − j.length() == 1

(·) Extra char = 1

abcd, abxcd, → 2

"a", "ab", "abc" → 3

"a", "ab", "abc", abcd, abxcd

1 2 3 4 5

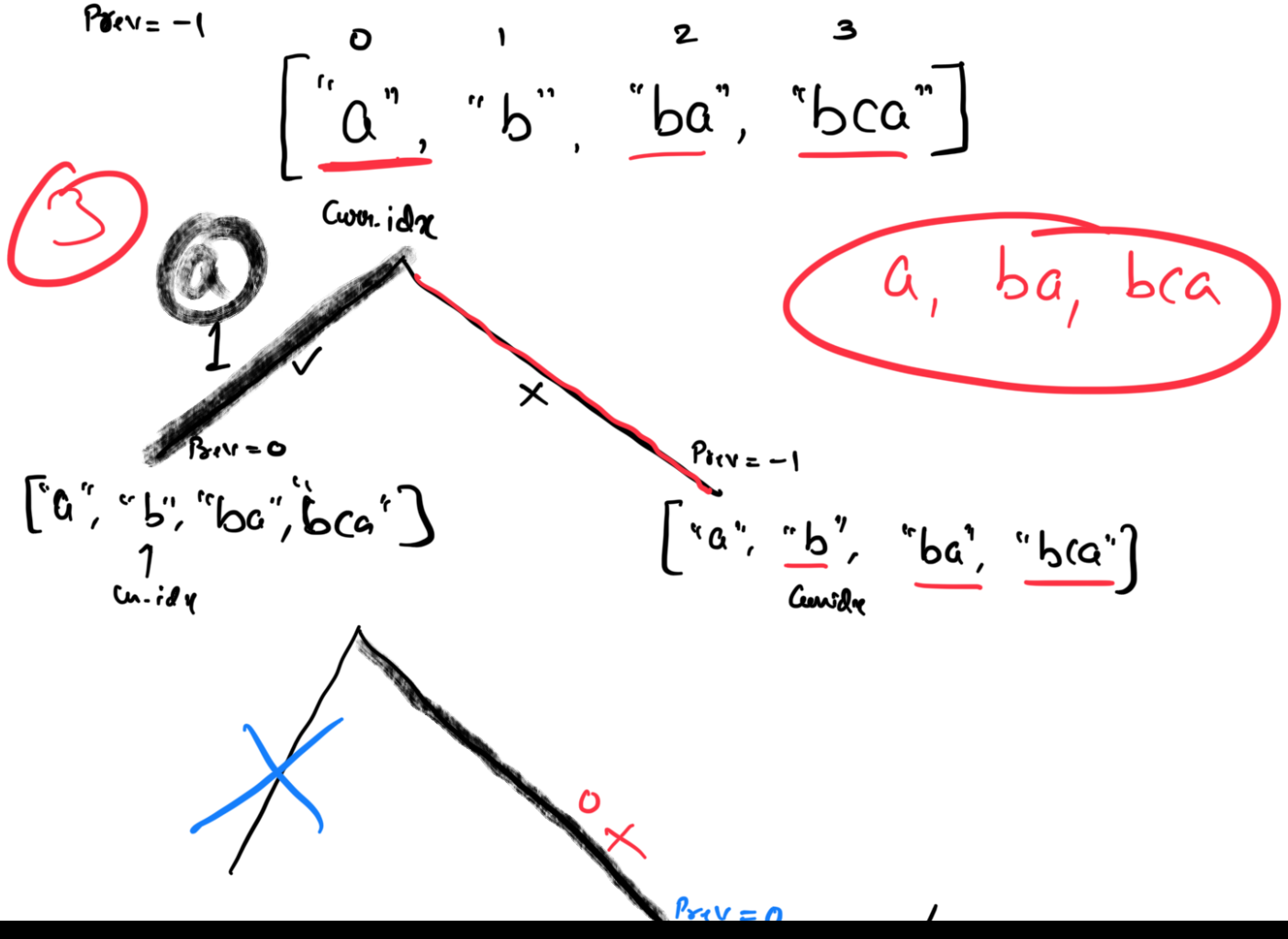
5

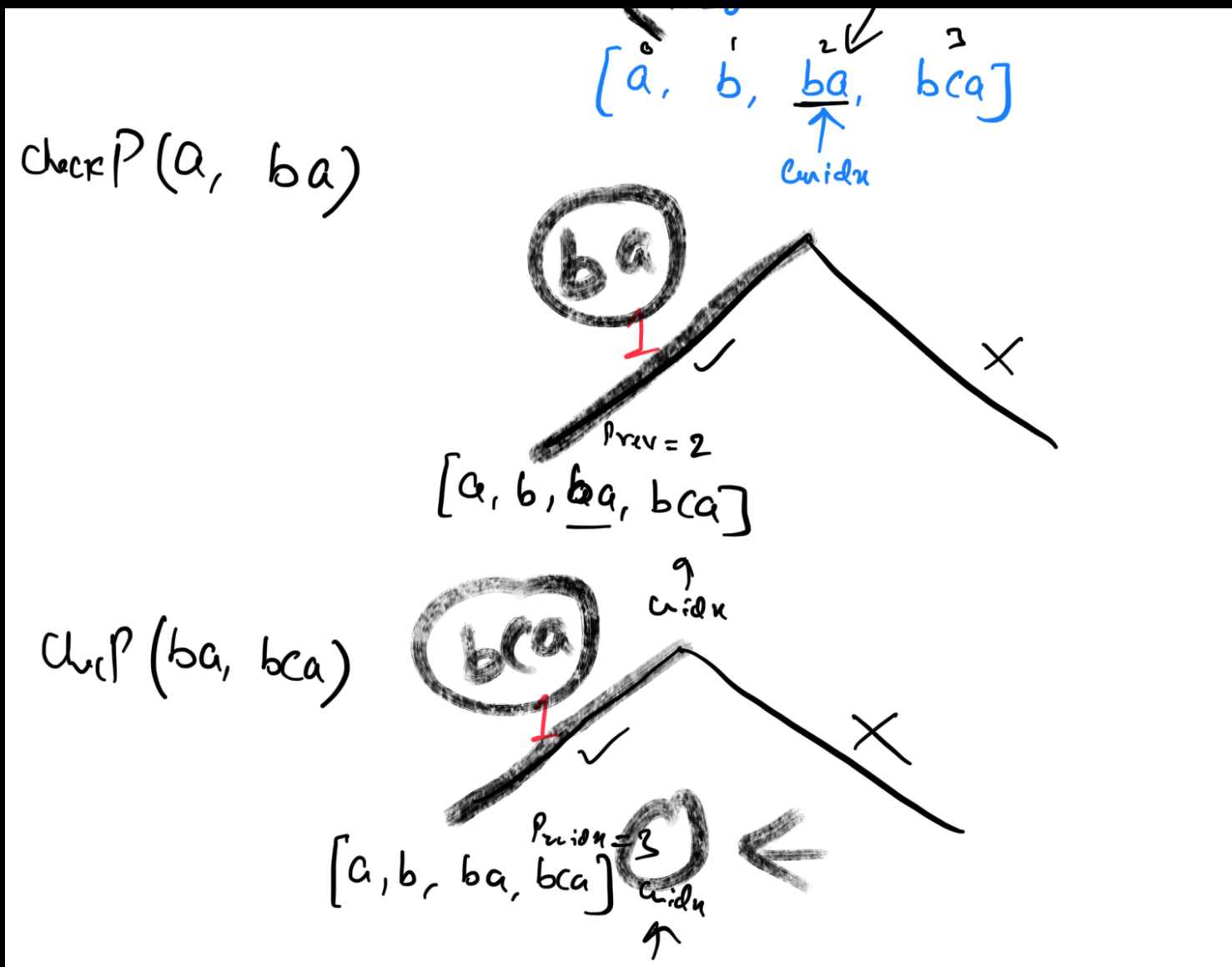
(•) Sorting. (based on length)

(•) Predecessor. (function).

length ↗ + Predecessor

DRY RUN ...



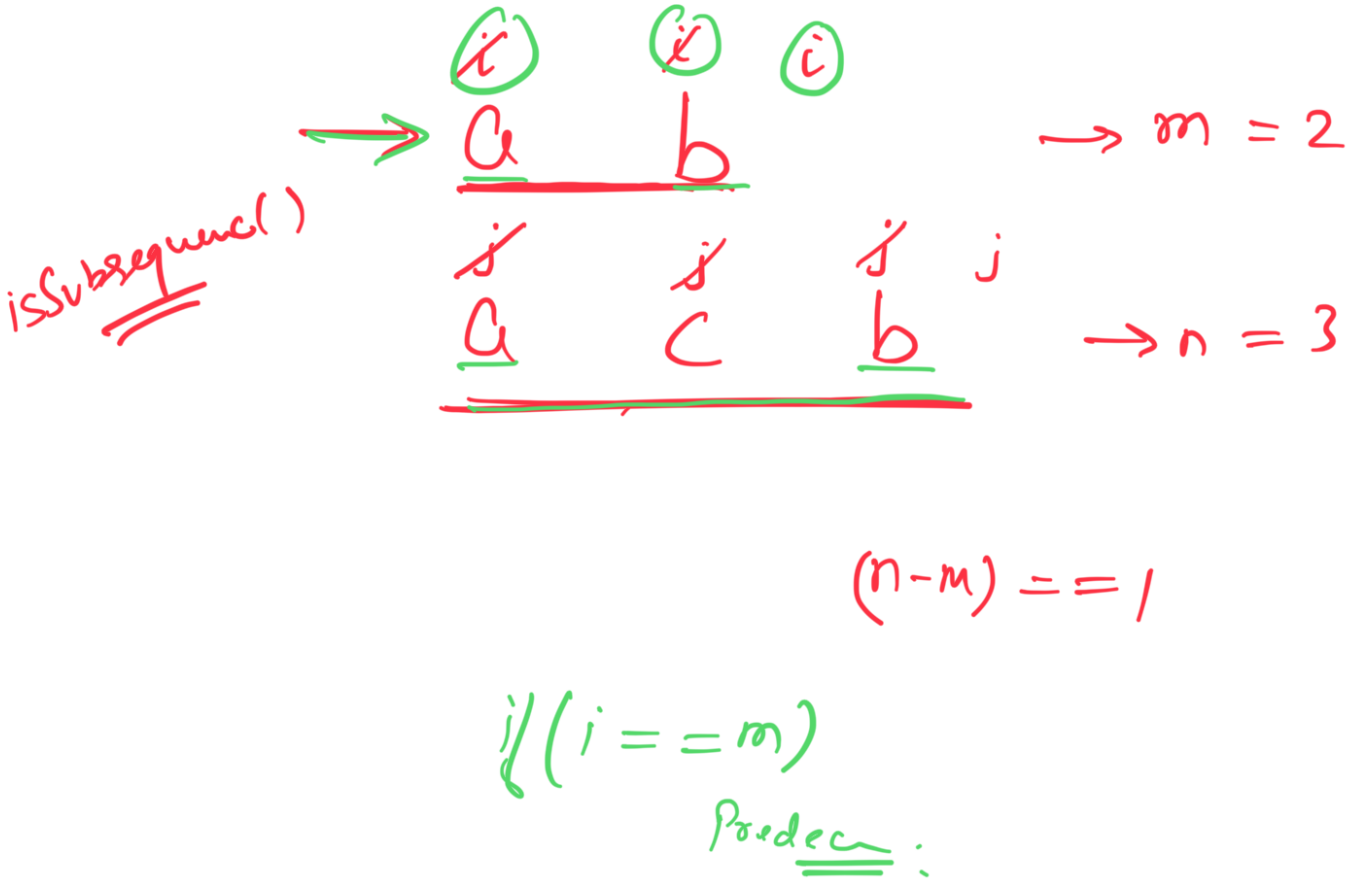


① Sorting ✓

② checkPredecessor ✓

③ LIS 의 code copy + Paste.

checkPredecessor("ab", "acb")



(n-m) == 1

if(i == m)

Predec: