Recursion Concepts & Qns ...

Motivation (भाषण) ...

"Be very careful on where you are spending your time today. This will impact your future. Use it wisely."

video

18

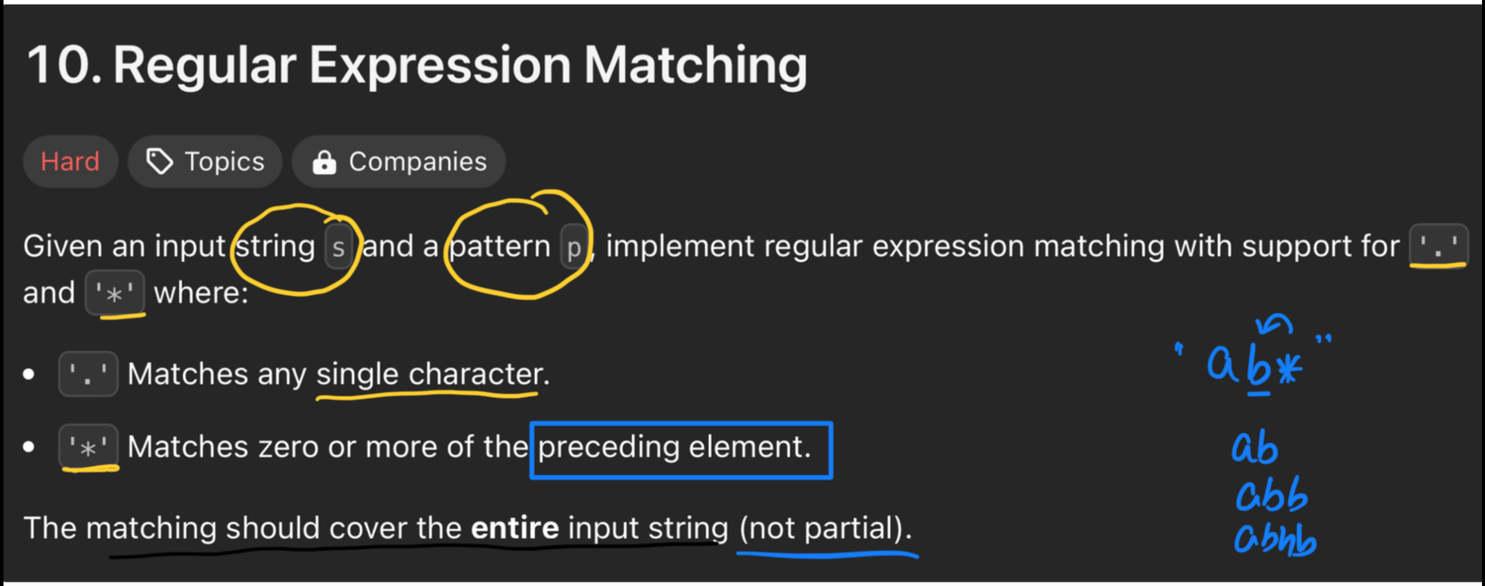
codestorywithMIK

Facebook
Instagram → codestorywithMIK

(Twitter) → CSwithMIK

codestorywithMIK ->

Company :- Google



10. Regular Expression Matching

Hard · Topics · Companies

Given an input string `s` and a pattern `p`, implement regular expression matching with support for `'.'` and `'*'` where:

- `'.'` Matches any single character.
- `'*'` Matches zero or more of the preceding element.

The matching should cover the entire input string (not partial).

"ab*"

ab
abb
abbb

Example :- S = "aa" , P = "a*"

Output = True

S = "ab" , P = ".*"

Output = True

Thought Process :-

① "ab" ".a" False.

② "aa" ".a" True

⇒ ③ "ab" ".*b"

Don't take it

(Recursion)

True

Take it

④ "abb" ".*b"

Don't take. ✗

"b" ".*b"

.*

"b" "b"

True.

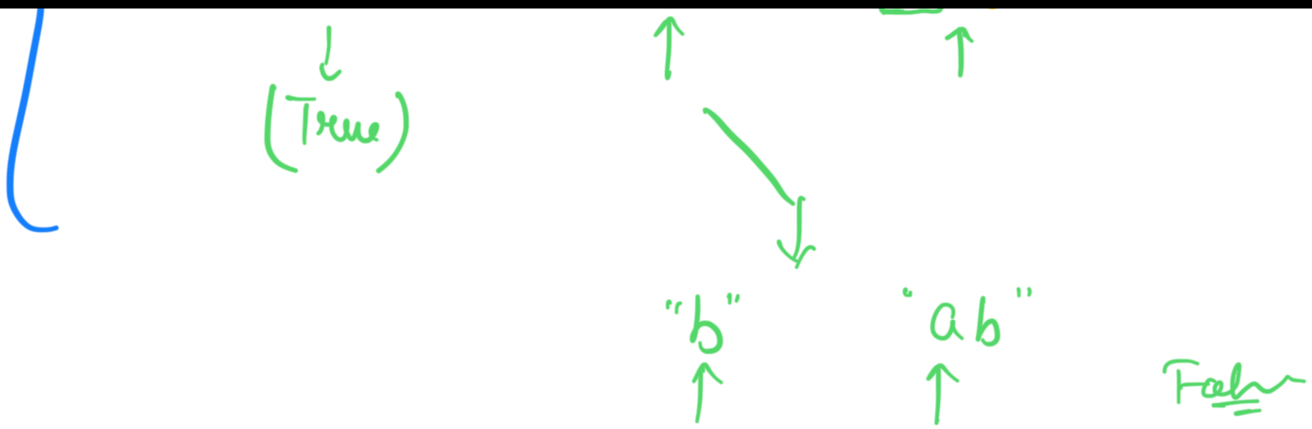
⑤ "ab" "a*ab"

take *

.*

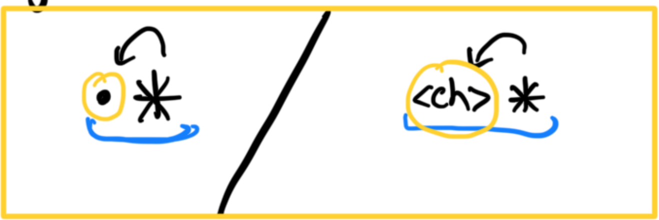
Char *

Don't take "b" "a*ab"



Summary:-

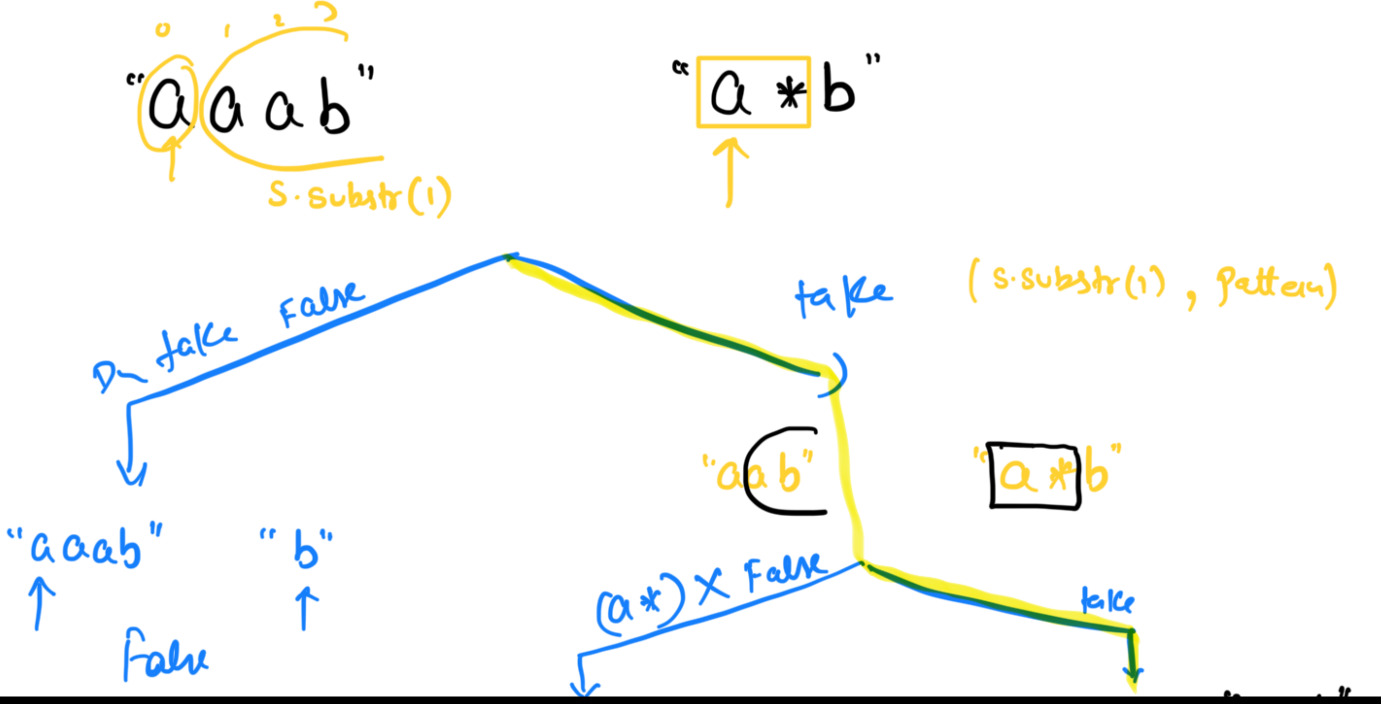
(•)

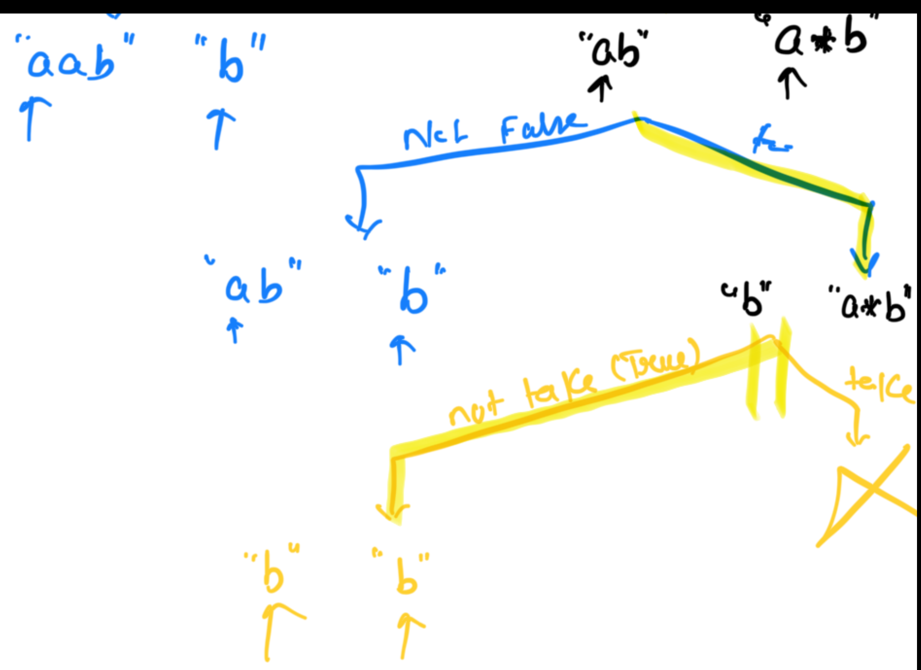


(•) 0 times लिया → pattern.substr(2)

(•) First check that (•) || <ch> == input[0]

Dry Run:-





Story Code:

```
bool Solve ( s , p) {

    if ( p.length() ==0) {
        if (s.length()==0) re True;
        retn False;
    }

    bool first_char_matched = p[0]==s[0] || p[0]=='.' ;

    if ( p[1] == '*') {

        bool not_take_* = Solve(s, p.substr(2));

        bool take_* = (p[0] == s[0] ||p[0]=='.' &&
solve (s.substr(1), p);

} else {

    return (p[0] == s[0] || p[0] == '.' && &
            solve (s.sub(1), p.sub(1)));

}
```

S.C = max depth of rec. → O(m)

T.C =

$$T(n) = 2T(n-1)$$

↓

$$= 2^n$$

T.C 2^n

Optimisation :-

i

"aba"

↑

j

". b ."

↑

$s[i] == p[j] \;||\; p[j] == \text{'.'}$

~~Substr~~ → i, j