Q1 ⇒ Given 2 string, find length of longest common subsequence LCS (s1, s2)

$S_1 : N$ & $S_2 : M$

Ex:

	0	1	2	3	4	5	6
S_1 :	a	b	b	c	d	g	f
S_2 :	b	a	c	h	e	g	f

acgf
bcgf } ans : 4

ex2 :

s: klagrip

s2: lgagkm

lgi : 3

LCS (s1 [0-6], s2 [0-6])

if (s1[6] == s2 [6])

1 + LCS (s1[0-5], s2 [0-5])

if (s1[5] == s2 [5])

1 + LCS [s1 [0-4], s2 [0-4])

if (s1 [4] == s2 [4])

	0	1	2	3	4
s1:	a	b	b	e	d
s2	b	a	c	h	e

abbcd abbe

bach bache

LCS (s1 [0-4], s2 [0-3])

if (s1[4] == s2 [3])

LCS [s1[0-3], s2[0-4]

if (s1 [3] == s2[4]

LCS [s1 [0,3), s2 [0-3])

LCS (s1 [0-3], s2 [0-3])

dp[i, j] → ending index of s1

j → s2

dp Expr :

```
if (s1[i] == s2[j])
        1 + dp[i-1, j-1]
else
    max(dp[i, j-1], dp[i-1, j])
```

Recursive Code

i.

0 a 0 b

0 abcd "" -1

0 1 2 3 a b c d , 0 1 2 3 b c d d ⇒ 3

L(3, 3)

abcd
bcdd

1 + L(2, 2)

abc
bcd

(ab, bcd) (abc, bc)

1 + L(2, 1)

(a, bcd) (ab, bc)

⇒ int dp[N][M] = -1

```
int LCS( s1, s2, i, j, dp[][])
{
    if (i == -1 || j == -1)   return 0;

    if (dp[i][j] == -1)
    {
        if (s1[i] == s2[j])
            dp[i][j] = 1 + LCS(s1, s2, i-1, j-1, dp)
        else
            dp[i][j] = max { LCS(s1, s2, i-1, j, dp),
                             LCS(s1, s2, i, j-1, dp
    }
    return dp[i][j];
}
```

TC: $O(N*M) \cdot O(1) \Rightarrow O(N*M)$

SC: $O(N*M)$

abdc

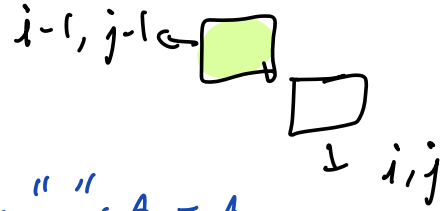
a b c d e

S_1 : MAICA

S_2 : IAIYAS

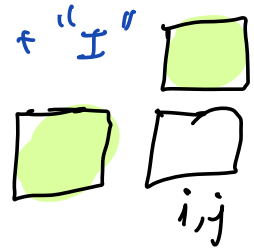
		I	A	I	Y	A	S
					M	M	
M	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1
I	1	1	2	2	2	2	2
C	1	1	2	2	2	2	2
A	1	2	2	2	3	3	3

```
if ( s1[i] == s2[j]
      1 + dp[i-1, j-1]
else
    max ( dp[i, j-1], dp[i-1, j] )
```



ans = " " + A = A

ans = "A" + "I"



S1: I

S2: MAIC

```
i = N-1 ; j = M-1
string ans = " "
while ( i >=0 && j >=0)
  {
      if ( s[i] == s[j])
        {
            ans = ans + s[i];
            i-- ; j--;
        }
      else
        { if( i>0 && dp[i][j] = dp[i-1][j]
            i--;
```

```
        else    j--;
    }
```

Q2 ⇒ Edit distance

Given 2 strings s1 & s2 min operations to be performed in s1 so that s1 becomes s2

In 1 operation of s1:

We can insert a char in s1 at any position

We can replace a char in s1 at any position, with any char

we can delete a char in s1 at any position

hurt
hut ⇒ 1

hurt
hust ⇒ 1

s1: d f a e l

s2: f g l

Ed(s1(0,4) , s2 (0, 2))

if (s1[4] == s2 [2])

⇓

Ed (s1 [0, 3], s2 [0, 1])

if (s[3] == s2[1])

Insert s1

s1: dfaeg

s2: fg

Ed(s1[0, 3], s2[0,0])

Replace s1

s1: dfa~~e~~g

s2: fg

Ed[s1(0,2), s2[0,0]

Delete s1

s: dfae

s2: fg

s1(0,2), s2(0,1)

dp(i, j) =

if (s1(i) == s2(j))

dp(i)(j) = dp(i-1, j-1)

else

dp(i)(j) = 1 + min { dp[i, j-1], dp(i-1, j-1], dp(i-1, j) }

dp(N](M) = -1

Break: 9:05

int Ed(s1, s2, i, j, dp()())

{

if (i == -1 && j == -1) return 0;

if (i == -1)

return j+1;

if (j == -1)

```
        return i+1;
    if (dp[i][j] == -1)
    {
        if (s1[i] == s2[j])
            dp[i][j] = ed(s1, s2, i-1, j-1, dp)
        else
            dp[i][j] = 1 + min { ed(s1, s2, i, j-1, dp),
                                 ed(s1, s2, i-1, j-1, dp),
                                 ed(s1, s2, i-1, j, dp) }
    }
    return dp[i][j]
}
```

Q3 ⇒ Regex Maching

// Given Text T & pattern P, check if both are same or Not.

T → In text it only contains alphabet.

P → with alphabet, it contains ?, *

Note P → It can have any number of ? & *

T: apple
P: a?*e
} → matching

T: apple
P: a*a?e
} → not matching

T: ant
P: a?**t

$\Rightarrow$ matching

T: " "
P: ****

$\rightarrow$ matching

T: " "
P: ?

$\Rightarrow$ X

T: aa
P: ????

0 1 2 3 4
apple
a*?e

RM(T[0,4], P[0,3])
if (T[4] == P[3])

$\Downarrow$

RM(T[0,2], P[0,2])
if (T[2] == P[2] || P[2] = "?")

$\Downarrow$

RM(T[0,2], P[0,1])
if (T[2] == P[1] || P[2] = "*"

ap
a*

app
a*

abb (without *)
a

RM(T[0, 2], P[0, 0])

ab
a*

RM(T[0,1], P[0,1])

dp Exp

```
if (T[i] == P[j] || P[j] == '?')
    dp[i][j] = dp[i-1, j-1],
else if (P[j] == '*')
    dp[i, j] = dp[i, j-1] ||
               dp[i-1, j]
else
    dp[i][j] = False,
```

	0	1	2	3	4
T ⇒	a	b	c	d	e
P ⇒	a	*	*	*	

T = " "
P = "* * * *"

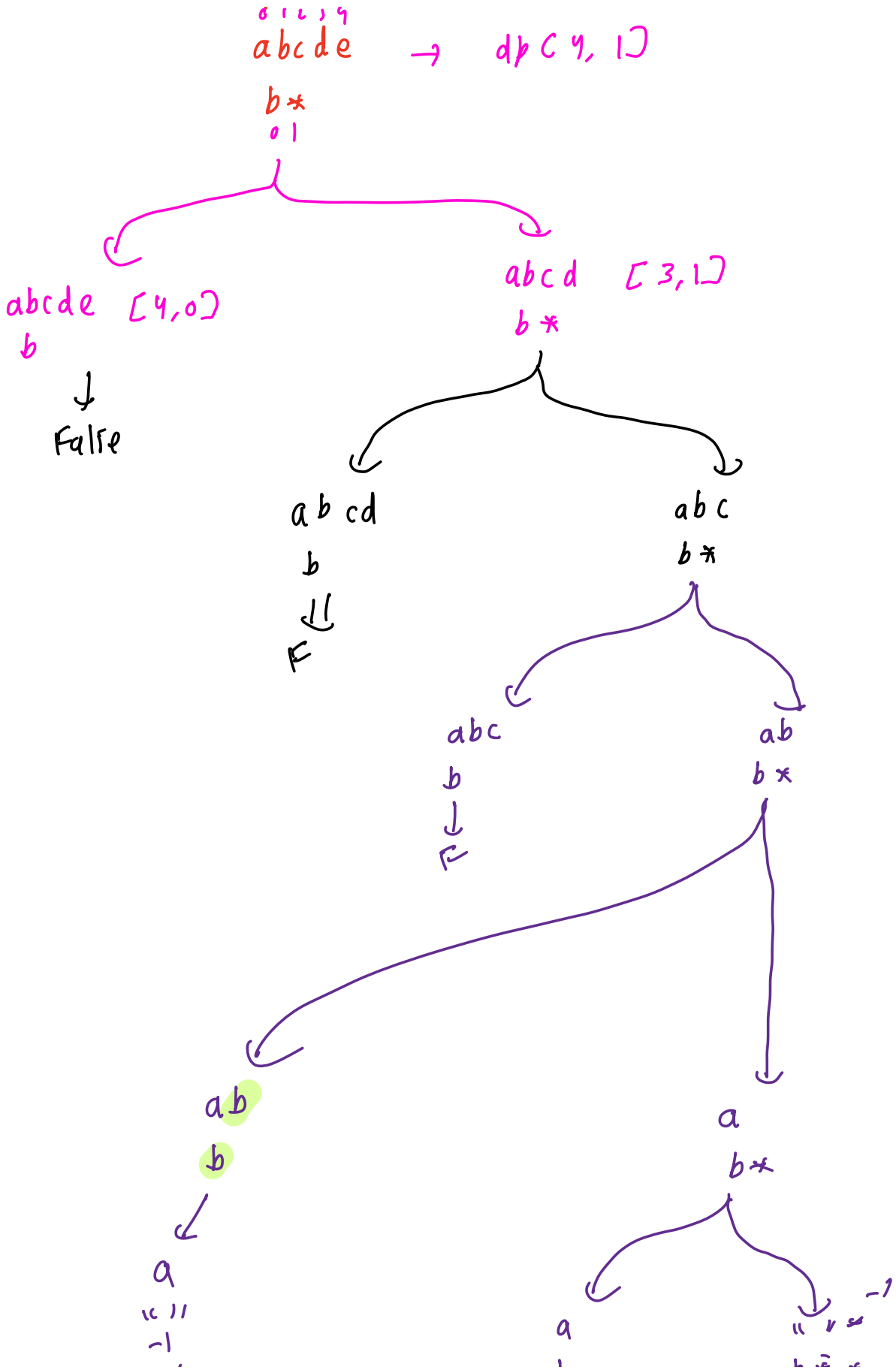
dp[4][3] =

abcde
a **

dp(4, 2)

abcd
a ***

dp(3, 3)


0 1 2 3 4
abcde
b*
0 1
dp[4, 1]
abcde [4,0]
b
False
abcd [3,1]
b*
abcd
b
F
abc
b*
abc
b
F
ab
b*
ab
b
a
a
b*
a
-1
a
-1

$$\left.\begin{array}{ll} T: & \text{" "} \\ b: & \text{" } b \times \text{ "} \end{array}\right\} \to F$$

$$\left.\begin{array}{ll} T: & \text{" "} \\ b: & \text{" } \times \ \times \text{ "} \end{array}\right\} \to T$$