Q given 2 strings, find the length of **longest common subsequence**

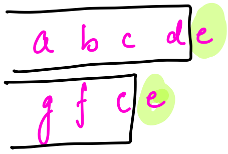
lowercase

S1: a b b c d g f
S2: b a e d e g f

a c d g f
ans = 5

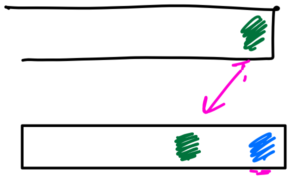
s1: k l a g r i p
s2: l g i g k m

lgi
ans = 3



a b c d
b c e f

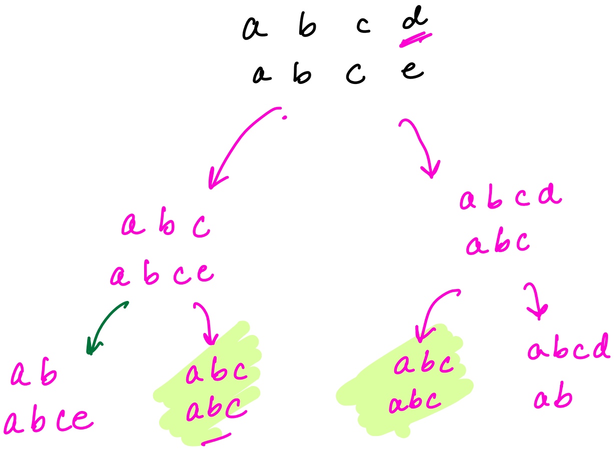
d & f can't coexist in the soln



LCS s1(0-n-1) & s2(0-m-1)

s1[n-1] == s2[m-1] → 1 + LCS((0-n-2), (0-m-2))

s1[n-1] != s2[m-1] → max (LCS(0-n-2, 0-m-1) , LCS(0-n-1, 0-m-2))

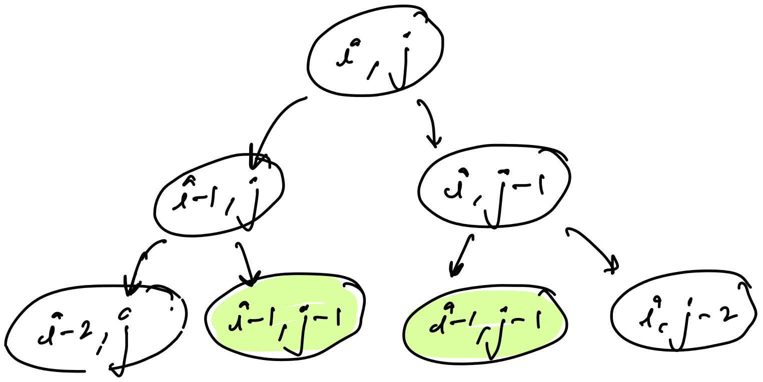


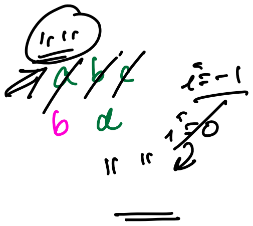
overlapping subproblems

$$\text{lcs}(i, j) = \begin{cases} 1 + \text{lcs}(i-1, j-1); & \text{if } s1[i] == s2[j] \\ \max(\text{lcs}(i-1, j), \ \text{lcs}(i, j-1)) & \text{else} \quad x \end{cases}$$

$\boxed{1+}$,
i, j

□





```
int    lcs[n][m];          // -1
                      → indices
findlcs( s1, s2, i, j)               0-i, 0-j
{
        if( i==-1 || j==-1) return 0;

        if( lcs[i][j] != -1) return lcs[i][j];

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

s1= KAIYA

s2= MAICA

dp[5][5]

		0	1	2	3	4
		M	A	I	C	A
K	0	0	0	0	0	0
A	1	0	1	1	1	1
I	2	0	1	2	2	2
Y	3	0	1	2	2	2
A	4	0	1	2	2	3

length

$i = n-1, j = m-1$

ans = 'AIA'

final ans = reverse(ans);

T.C: $O(n*m)$

S.C: $O(n*m)$

better opti ⇒ $O(m)$

#2 Edit distance

s1 → s2 can't change s2

few operations

insert
delete
replace

Find minimum cost

cost associated

insert ✓
delete ✓
replace

char

s1 = a c b
s2 = a b c

abc

= 2

cost i = 2
cost d = 2
cost r = 3

replace
e

s1 = a b c d - delete
s2 = a b e

2

⇒ 5

6

g replace 3

s1 = a c d x y - delete
s2 = a b c g x

2

7

a b c d
a e d

mincost (0-n-1 , 0-m-1)

s1[n-1] == s2[m-1]

mincost(n-2, m-2)

s1[n-1] != s2[m-1]

min

insert
costi
+
mincost (n-1, m-2)

a b c d g d
a b d

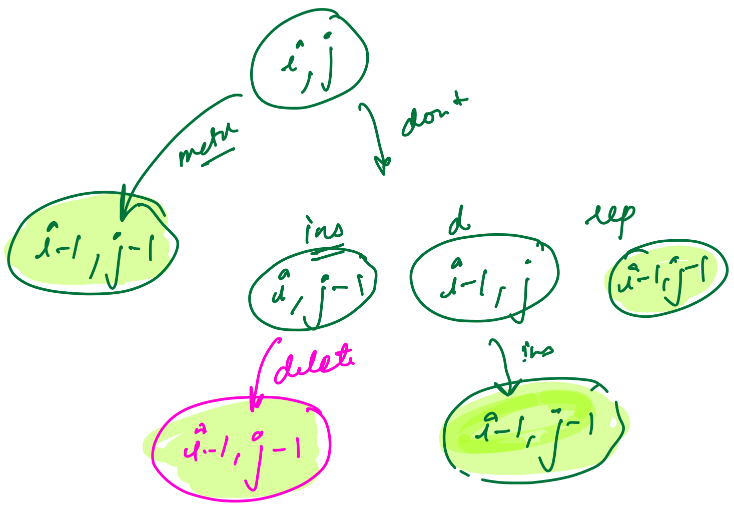
d
costd
+
mincost(n-2, m-1)

a b c d x
a b d

replace
costr
+
mincost(n-2, m-2)

d

a b c d
a b d



i=1

0 1

s1 = "ab"

s2 = ""

s1 = ""

s2 = "ab"

```
if( i ==-1 && j = =-1) return 0;
else if( i | =-1 && j==-1) return costd * (i+1);
else j!=-1 && i==-1 return costi * (j+1);
```

$$dp[i][j] = \begin{cases} dp[i-1][j-1] & \text{if } s1[i]==s2[j] \\ \min(\,cost\,i + dp[i][j-1], \\ \quad cost\,d + dp[i-1][j], \\ \quad cost\,r + dp[i-1][j-1]). & \times \end{cases}$$

[n+1][m+1]

	" "	a	b	c	f
" "	0	1* icost	2*c	3*c	4*c.
a	1*costd				
b	2*costd				
e	⋮				
f	|				

#3 Wildcard pattern match

Regex expressn

s: a b a c d
p: a b a c d

'?' → replace with exactly one characte

p: a ? a ? d
s: a b b a d

patter → ? *

s: a b b a c
p: a * c

'*' ⇒ replace with sequence of 0 or more characters
↓
any

p: x * z * *
s: x b b z z c

if string matches patter or not ?

*
↓
a
ab
abc
abbc

abce
a?ce

a b c e
a ? c ?

check(n-1, m-1) ③ → s[n-1] != pat[m-1] return false

① (s[n-1] == pat[m-1] || pat[m-1]=='?')
check(n-2, m-2)

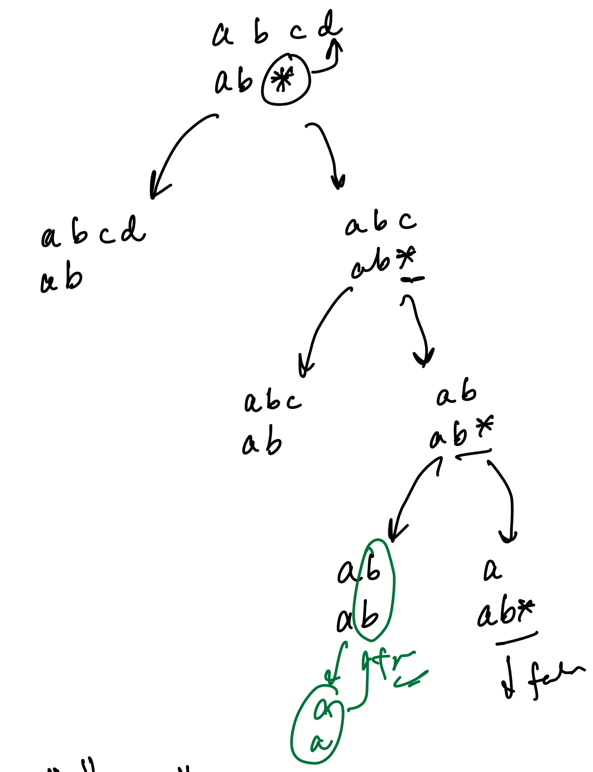
② pat[m-1]=='*'

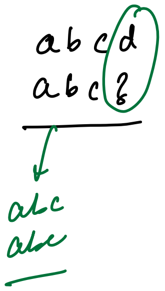
replace 0 element → check(n-1, m-2)

replace → check (n-2, m-1)

0 1

abcde
ab?d*





s=" "
p="****"

```
                  && j==-1
if(   i == -1  (&& j == -1)   return true;
else if(  i == -1 &&   return false;
else if(  j == -1) && || if only * are in pattern return true
       && i==-1              else return false;
```

$$dp[i][j] = \begin{cases} \Downarrow\ dp[i-1][j-1] & \text{if } (s[i]==p[j] \ \|\ p[j]=='?') \\ dp[i][j-1] \ \|\ dp[i-1][j] & \text{else if } (p[j]=='*') \\ \text{false;} & \text{else} \end{cases}$$