1. 부분 수열 : 부분 수열이란 서로 겹치는 수열(문자열)을 의미함

\중간에 건너 뛰어도 되지만 순서대로 지나가면서 겹치는 문자열들의 모임\

2. LCS : 부분 수열 중에서 가장 긴 것

ACAYKP 와 CAPCAK -> ACAK

3. Brute-Force로 풀기(무식하게) : X(X\_1~X\_m)의 모든 부분 서열의 개수 -> O(2^m)

4.

0) X = x\_1~x\_m

Y = y\_1~y\_n

Z = z\_1~z\_k

이렇게 있을 때 (Z는 최장 부분수열이라면)

1) z\_k = x\_m = y\_n

\둘이 같다면 z\_k가 정해짐\

z\_k-1 = LCS(X\_m-1, Y\_n-1)

\나머지 에서 z\_k-1를 다시 구하면 됨\

2) x\_m != y\_n

\둘이 공통 부분 수열이 아니라면 - > 경우 2가지\

ㄱ) z\_k != x\_m : Z = LCS(X\_m-1, Y\_n)

\z\_k가 x\_m이 아닐때\

ㄴ) z\_k != y\_n : Z = LCS(X\_m, Y\_n-1)

\z\_k가 y\_n이 아닐때\

3) 재귀적으로 정의하기

ㄱ) c(i, j) : 수열 X\_i와 Y\_j의 최장공통부분수열의 길이

ㄴ) 베이스케이스 : i = 0 또는 j = 0 이면 c(i, j) = 0

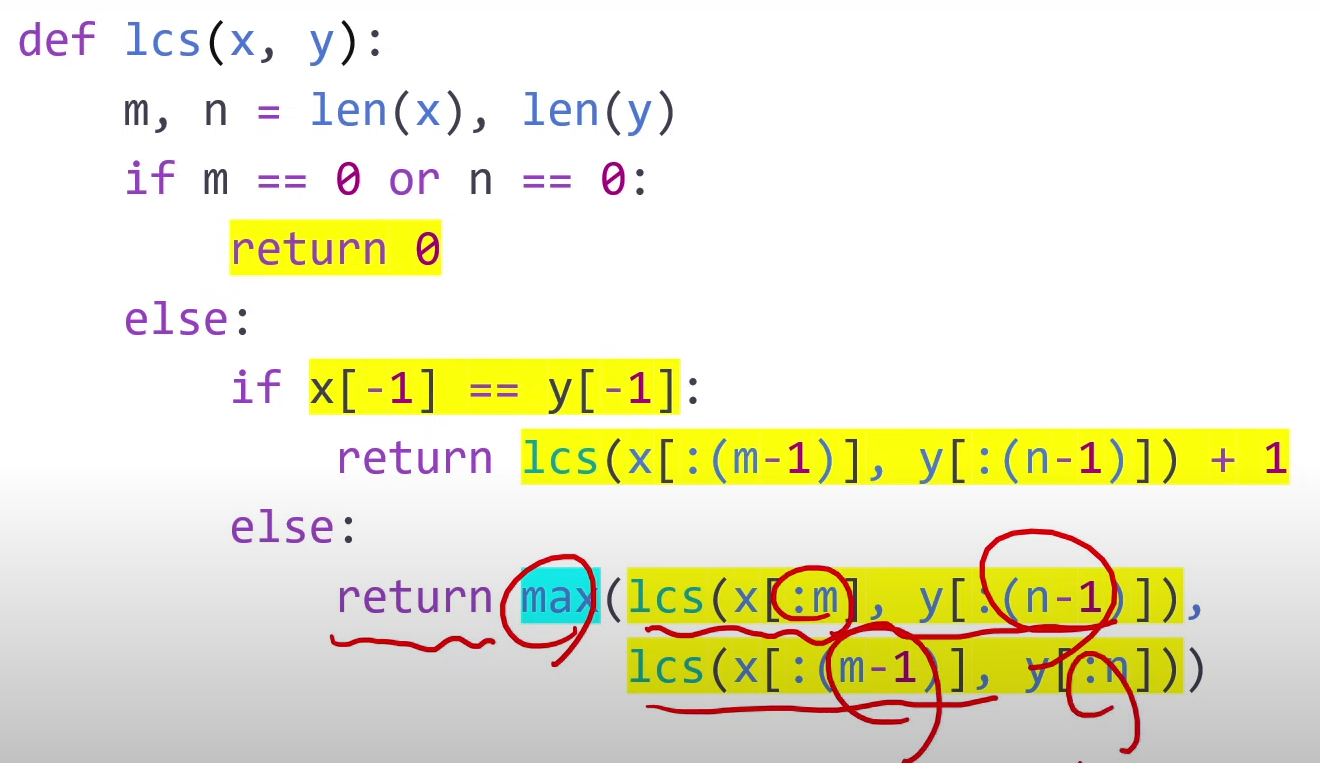
ㄷ) 재귀조건(점화식)

a. x\_i = y\_j 이면 c(i, j) = c(i-1, j-1) + 1

\이미 구한 공통부분수열 1개는 제외하고 나머지 X\_i-1과 Y\_j-1에서 공통부분수열 찾기\

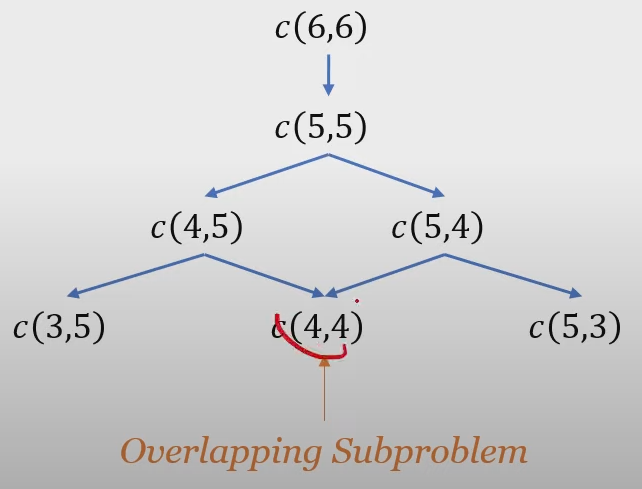
b. x\_i != y\_j 이면 c(I, j) = max{c(i, j-1), c(i-1, j)}

\각자 하나씩 빼서 공통부분수열의 길이를 찾아보고, 더 긴쪽을 선택하면 됨\



\* 중복 부분 문제

1) 'c(i, j) = 0' 이거나 'c(i, j) = c(i – 1, j – 1) + 1' 이거나 'c(i, j) = max{c(i, j-1), c(i-1, j)}

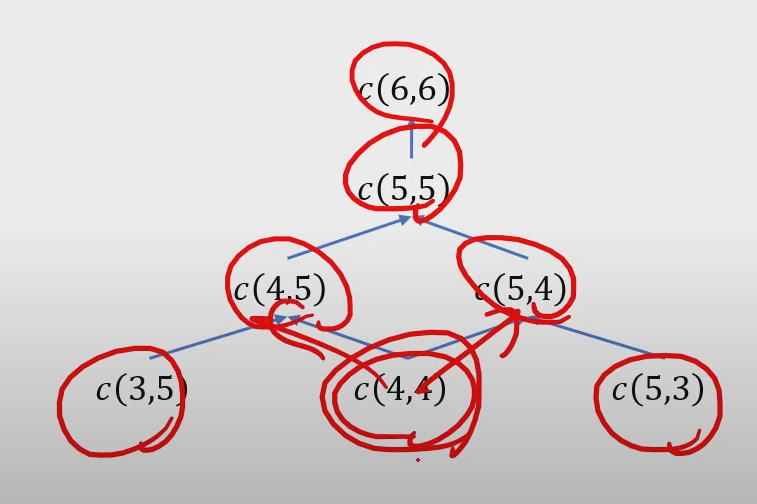


c(4,4)는 중복 부분 문제가 됨 - > 시간복잡도가 O(2^n)

\재귀적으로 갈라지다보면 c(4,4)처럼 중복되는 부분문제가 생김\

\* 동적계획법 풀이

\*\* 저런 중복 부분 문제가 있으면 해결해줄 수 있는 방법 -> 메모이제이션



\상향식으로 생각해보면 밑에서 c(4,4)를 테이블이 저장해둬서 중복될 때마다 메모이제이션 해둔걸 사용하면 됨\

# 9251 LCS

S1 = list(input())

S2 = list(input())

# 문자열을 입력받아서 각 문자별로 리스트에 저장

len1 = len(S1)

len2 = len(S2)

# 각 길이를 저장

dp = [[0]\*(len2 + 1) for \_ in range(len1+1)]

# 메모이제이션할 공간 만들기(하나씩 더 공간을 만드는 이유 : 처음 시작하는 0000으로 시작하는 세로와 가로를 세팅하기 위해서)

# print(dp)

for i in range(1,len1 + 1) :

    for j in range(1,len2 + 1) :

        if S1[i-1] == S2[j-1] :

            dp[i][j] = dp[i-1][j-1] + 1

        else :

            dp[i][j] = max(dp[i-1][j],dp[i][j-1])

# 표로 설명

# print(dp)

print(dp[len1][len2])