Approach:

1. Similar to LCS
2. In LCS we take longest common subseq between 2 strings:

That is

“rabbit” and “rabbbit”

LCS -> “rabbit”

Here we need to check how many times we can get the subsequence “rabbit” (t) from another string “rabbbit” (s)

Therefore, we already have an LCS, just figure out how many times LCS occurs!

1. Base condition:
   1. If t is empty: make s empty-> can form a valid subsequence of t.

So if t=””, count = 1

* 1. If s is empty: cannot form a subseq,

So if s=”” count = 0

* 1. If s[i] matches t[j]:

2 possibilities:

1. Skip/ don’t consider current letter in s, maybe ‘s’ has other occurrences: (i+1),(j)
2. Consider current letter, and move to next letter: (i+1),(j+1)

**Note: add both these possibilities, to get subsequences**

d) s[i] != t[j]

Skip i check if next letter in s matches current letter of t(the idea is to find a subsequence of t in s, so don’t skip any letter in t, until a match is found)

(i+1),(j)

**Code**

m = len(s)

n = len(t)

dp=[]

for i in range(m+1):

dp.append([])

for j in range(n+1):

dp[i].append(-1)

for i in range(m+1):

for j in range(n+1):

# if 's' is empty, no match

if j==0:

dp[i][j] = 1

# if 't' is empty, remove all from 's', matches empty t

elif i==0:

dp[i][j] = 0

elif s[i-1] == t[j-1]:

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

else:

# no match

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

return dp[m][n]