Practical 05

Name:- Mo Atif Azmi

Batch:-A4\_B4\_51

Date:-15-10-2025

Aim: Implement a dynamic algorithm for Longest Common Subsequence (LCS) to find the length and LCS for DNA sequences.

TASK 01:- Find the similarity between the given X and Y sequence.

X=AGCCCTAAGGGCTACCTAGCTT

Y= GACAGCCTACAAGCGTTAGCTTG

Output: Cost matrix with all costs and direction, final cost of LCS and the LCS.

Length of LCS=16

Code:-

def LCS\_Length(X,Y):

m, n = len(X),len(Y)

C = [[0]\*(n+1) for \_ in range(m+1)]

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

for j in range(1, n+1):

if X[i-1] == Y[j-1]:

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

else:

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

return C[m][n]

print(LCS\_Length("X=AGCCCTAAGGGCTACCTAGCTT","GACAGCCTACAAGCGTTAGCTTG"))

Output:-



TASK-2: Find the longest repeating subsequence (LRS). Consider it as a variation of the longest common subsequence (LCS) problem.

Let the given string be S. You need to find the LRS within S. To use the LCS framework, you effectively compare S with itself. So, consider string1 = S and string2 = S.

Example:

AABCBDC

LRS= ABC or ABD

CODE:-

def longest\_repeating\_subsequence(string):

n = len(string)

table = [[0 for \_ in range(n+1)] for \_ in range(n+1)]

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

for j in range(1, n+1):

if string[i-1] == string[j-1] and i != j:

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

else:

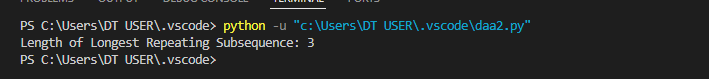
table[i][j] = max(table[i-1][j], table[i][j-1])

return table[n][n]

S = "AABCBDC"

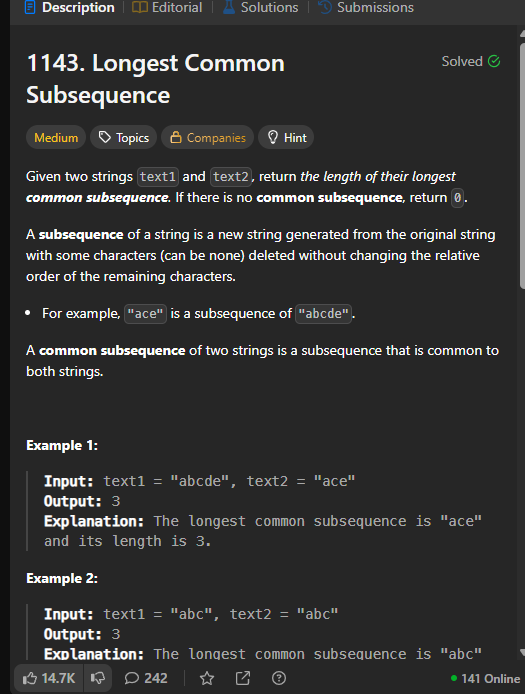
print("Length of Longest Repeating Subsequence:", longest\_repeating\_subsequence(S))

OUTPUT:-

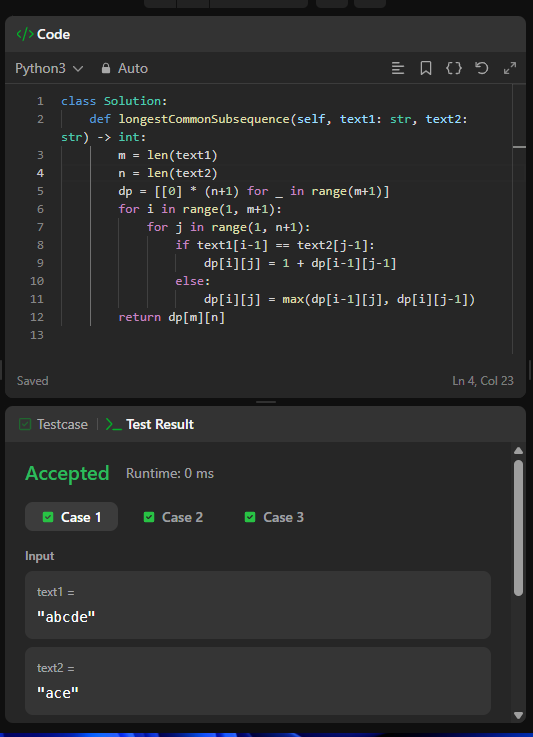


LeetCode Assessment:

Problem:-



Code:-



Submission:-

