**Practical No.5**

|  |  |
| --- | --- |
| **Name** | **Aditya Dubey** |
| **Batch** | **A6-B3** |
| **Roll no** | **36** |

**Aim: Implement a dynamic algorithm for Longest Common Subsequence (LCS) to find the**

**length and LCS for DNA sequences.**

**Problem Statement:**

**(i) DNA sequences can be viewed as strings of A, C, G, and T characters, which**

**represent nucleotides. Finding the similarities between two DNA sequences are an**

**important computation performed in bioinformatics.**

**[Note that a subsequence might not include consecutive elements of the original sequence.]**

**TASK 1: Find the similarity between the given X and Y sequence.**

**X=AGCCCTAAGGGCTACCTAGCTT**

**Y= GACAGCCTACAAGCGTTAGCTTG**

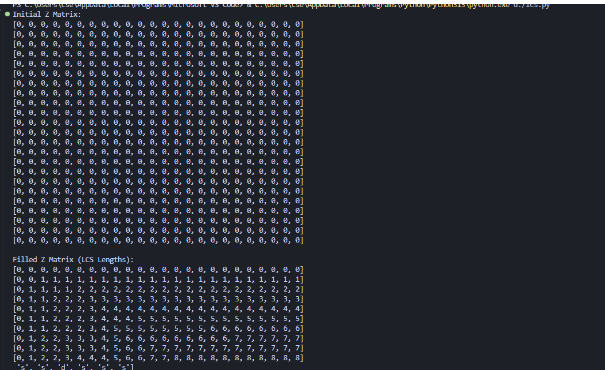
**Output: Cost matrix with all costs and direction, final cost of LCS and the LCS.**

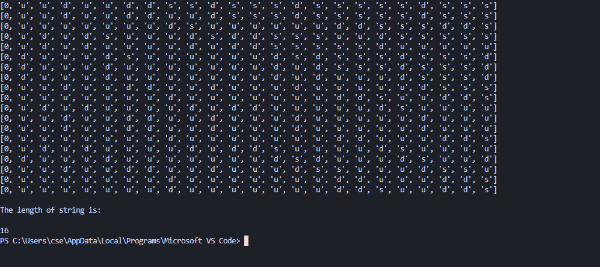
**Length of LCS=16**

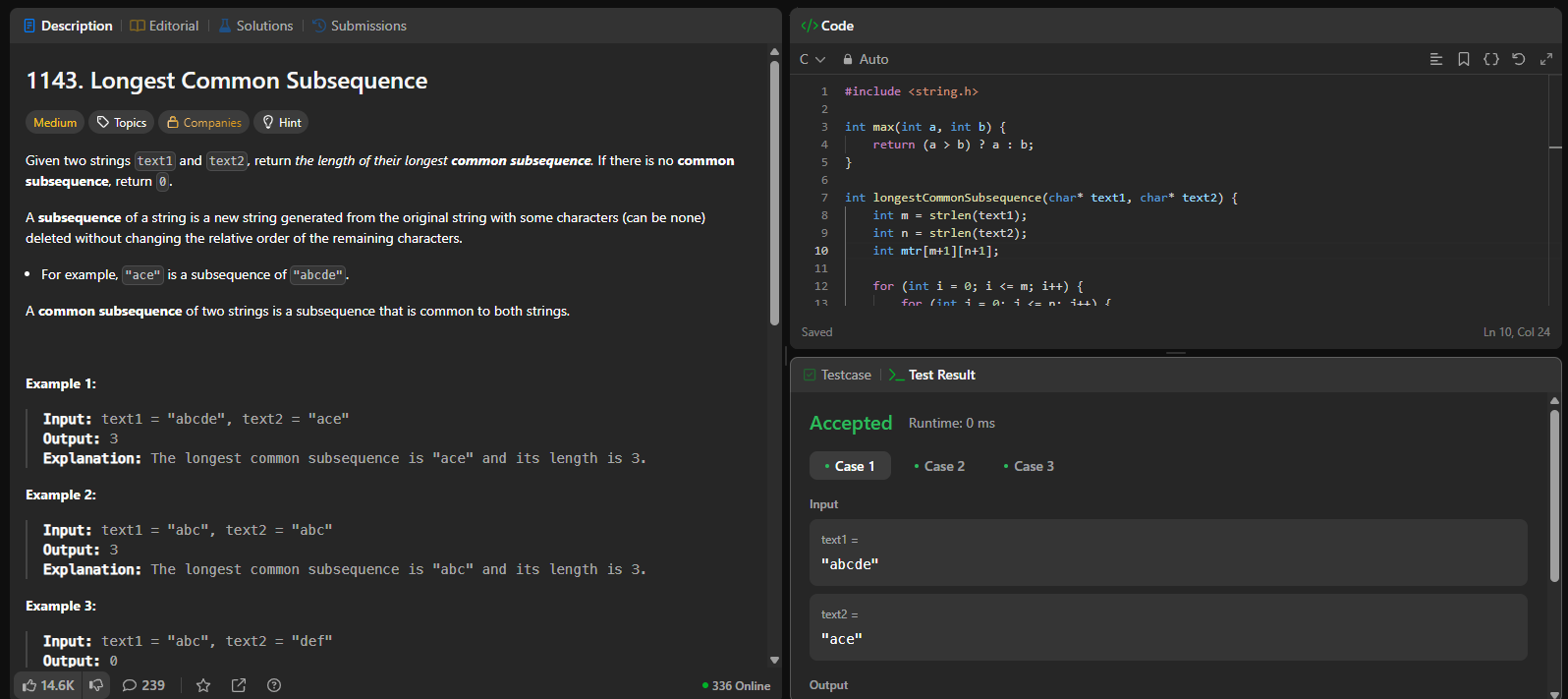
**CODE:-**

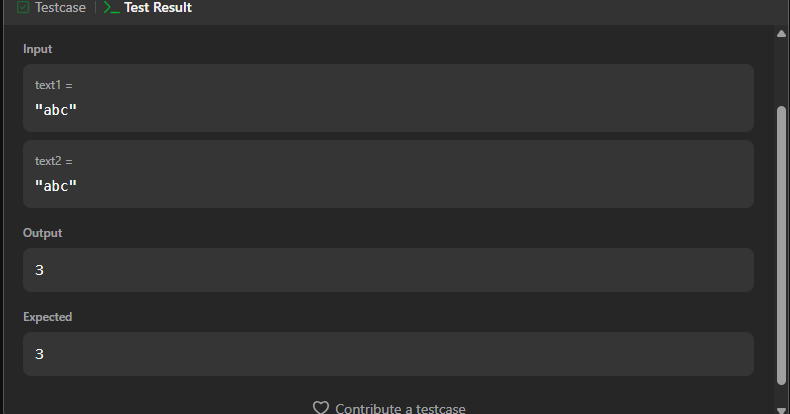
X = "AABC"  
Y = "ABBC"  
  
m = len(X)  
n = len(Y)  
  
Z = []  
D = []    
  
for i in range(m + 1):  
    row = []  
    for j in range(n + 1):  
        row.append(0)    
    Z.append(row)  
  
for i in range(m + 1):  
    row = []  
    for j in range(n + 1):  
        row.append(0)    
    D.append(row)  
  
print("Initial Z Matrix:")  
for row in Z:  
    print(row)  
  
for i in range(1, m + 1):    
    for j in range(1, n + 1):  
        if X[i - 1] == Y[j - 1]:    
            Z[i][j] = Z[i - 1][j - 1] + 1    
            D[i][j] = 'd'    
        else:  
            if Z[i - 1][j] >= Z[i][j - 1]:  
                Z[i][j] = Z[i - 1][j]    
                D[i][j] = 'u'    
            else:  
                Z[i][j] = Z[i][j - 1]  
                D[i][j] = 's'    
  
print("\nFilled Z Matrix (LCS Lengths):")  
for row in Z:  
    print(row)  
  
print("\nDirection Matrix (u = up, s = side, d = diagonal):")  
for row in D:  
    print(row)

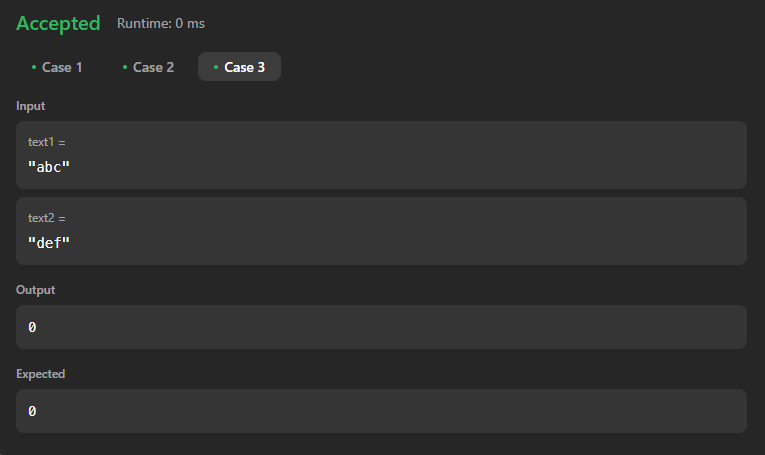
**OUTPUT:**

****

****

**LEETCODE:  
**

****

****

**Problem Statement :**

**LRS (Longest Repetitive Sequence)**

** The Longest Repeating Subsequence (LRS) of a string is the longest subsequence (not**

**substring) that appears at least twice in the string, and the two subsequences must not have**

**the same character index in both occurrences.**

** It is very similar to Longest Common Subsequence (LCS), but here we compare the string**

**with itself and ensure that the same index is not reused.**

**Input:**

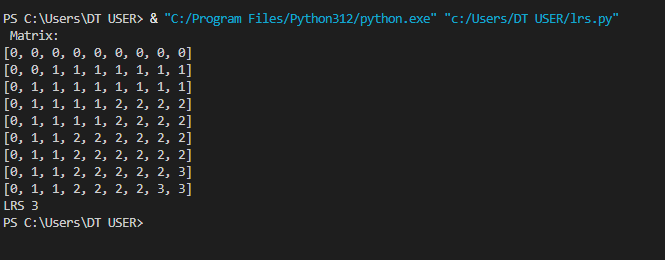
**s1=”AABEBCDD”;**

**s2=”AABEBCDD”;**

**CODE:**

def LRS(a, b):  
    n = len(a)  
    m = len(b)  
     
    c = [[0] \* (m + 1) for \_ in range(n + 1)]  
     
    for i in range(1, n + 1):  
        for j in range(1, m + 1):  
            if a[i - 1] == b[j - 1] and i != j:  
                c[i][j] = 1 + c[i - 1][j - 1]  
            else:  
                c[i][j] = max(c[i - 1][j], c[i][j - 1])  
     
    print(" Matrix:")  
    for row in c:  
        print(row)  
     
    return c[n][m]  
  
a = "AABEBCDD"  
b = "AABEBCDD"  
result = LRS(a, b)  
  
print(f"LRS {result}")

**OUTPUT:**

****