Rishit Saxena A5-B2-29

Practical 5

Code a = input("Enter first DNA sequence:

") b = input("Enter second DNA sequence:

")

m = len(a) n

= len(b)

M = [[{'v': 0, 'd': ''} for j in range(n+1)] for i in range(m+1)]

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

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

if a[i-1] == b[j-1]:

M[i][j]['v'] = M[i-1][j-1]['v'] + 1

M[i][j]['d'] = 'd' else:

if M[i-1][j]['v'] >= M[i][j-1]['v']:

M[i][j]['v'] = M[i-1][j]['v']

M[i][j]['d'] = 'u' else:

M[i][j]['v'] = M[i][j-1]['v']

M[i][j]['d'] = 's'

def bt(M, a, i, j): if

i == 0 or j == 0:

return "" if

M[i][j]['d'] == 'd':

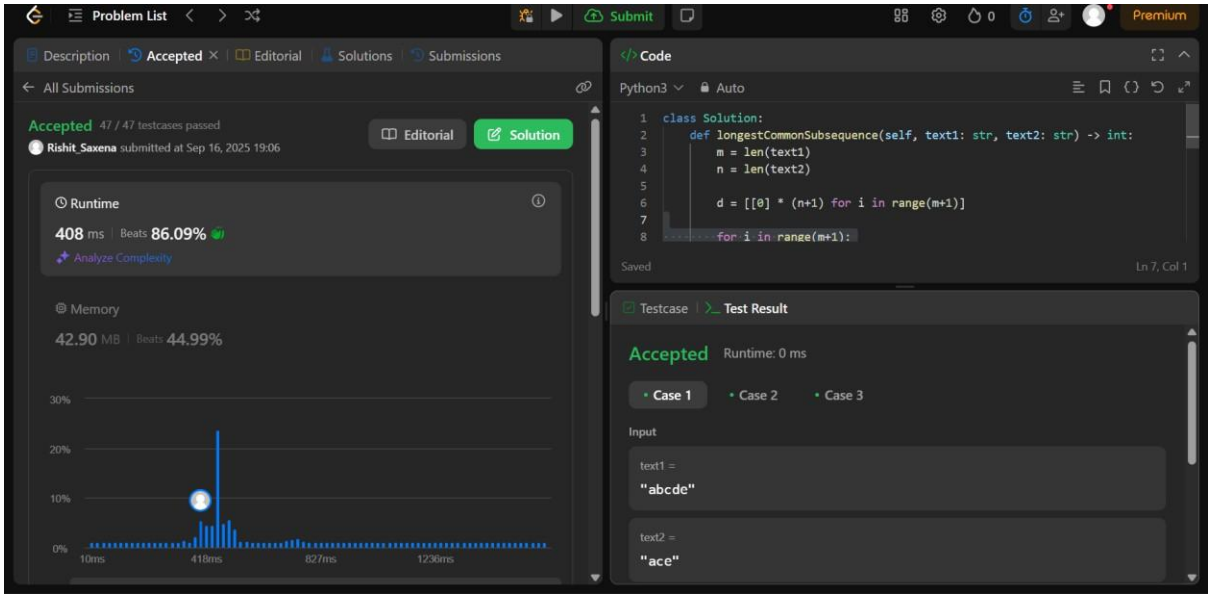
return bt(M, a, i-1, j-1) + a[i-1]

elif M[i][j]['d'] == 'u':

```
        return bt(M, a, i-1, j)
    else:
        return bt(M, a, i, j-1)

res = bt(M, a, m, n)

print("\nCost Matrix (values):")
for i in range(m+1):    for j in
range(n+1):
        print(M[i][j]['v'], end=" ")
print()

print("\nDirection Matrix:")
for i in range(m+1):    for j
in range(n+1):
        print(M[i][j]['d'] if M[i][j]['d'] != '' else '-', end=" ")
print()

print("\nLength of LCS:", M[m][n]['v']) print("LCS:",
res)
```


Enter first DNA sequence: AGCCCTAAGGGCTACCTAGCATT
Enter second DNA sequence: GACAGGCCTACAGGCTTAGCTTG
Cost Matrix (values):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
0 1 1 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
0 1 1 2 2 2 2 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
0 1 1 2 2 2 2 3 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5
0 1 1 2 2 2 2 3 4 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6
0 1 2 2 3 3 3 3 4 5 6 6 6 6 6 6 6 7 7 7 7 7 7 7
0 1 2 2 3 3 3 3 4 5 6 6 7 7 7 7 7 7 7 7 7 7 7 7
0 1 2 2 3 4 4 4 4 5 6 6 7 8 8 8 8 8 8 8 8 8 8 8
0 1 2 2 3 4 5 5 5 5 6 6 7 8 9 9 9 9 9 9 9 9 9 9
0 1 2 2 3 4 5 5 5 5 6 6 7 8 9 9 9 9 9 10 10 10 10 10
0 1 2 3 3 4 5 6 6 6 6 7 7 8 9 10 10 10 10 10 11 11 11 11
0 1 2 3 3 4 5 6 6 7 7 7 7 8 9 10 11 11 11 11 11 12 12 12
0 1 2 3 4 4 5 6 6 7 8 8 8 8 9 10 11 11 12 12 12 12 12 12
0 1 2 3 4 4 5 6 7 7 8 9 9 9 9 10 11 11 12 12 13 13 13 13
0 1 2 3 4 4 5 6 7 7 8 9 9 9 9 10 11 11 12 12 13 13 13 13
0 1 2 3 4 4 5 6 7 8 8 9 9 9 9 10 11 12 12 12 13 14 14 14
0 1 2 3 4 4 5 6 7 8 9 9 10 10 10 10 11 12 13 13 13 14 14 14
0 1 2 3 4 5 5 6 7 8 9 9 10 11 11 11 11 12 13 14 14 14 14 15
0 1 2 3 4 5 5 6 7 8 9 10 10 11 11 12 12 12 13 14 15 15 15 15
0 1 2 3 4 5 5 6 7 8 9 10 11 11 11 12 12 12 13 14 15 15 15 15
0 1 2 3 4 5 5 6 7 8 9 10 11 11 11 12 13 13 13 14 15 16 16 16
0 1 2 3 4 5 5 6 7 8 9 10 11 11 11 12 13 14 14 14 15 16 17 17
Direction Matrix:
- -
- u d s d s s s s s d s d s s s s s d s s s s s
- d u u u d d s s s s s s d d s s s s d s s s d
- u u d s u u d d s s d s s s d s s s s d s s s
- u u d u u u d d s s d s s s d s s s s d s s s
- u u d u u u d d u u d s s s d s s s s d s s s
- u u u u u u u u d s u u u u u d d s s s d d s
- u d u d s s u u u d s d s s s u u d s s s s s
- u d u d u u u u u d u d s s s s s d u u u u u
- d u u u d d s u u u u u d d s s s s d s s s d
- d u u u d d s s u u u u u d d s s s s d s s s d
- d u u u d d u u u u u u u d d u u u u d s s s d
- u u d u u u d d s u d u u u d s s s u d s s s
- u u u u u u u u d s u u u u u d d s s u d d s
- u d u d u u u u u u d s d u u u u u d s s u u u
- u u d u u u d d u u d s s u d u u u u d s s s
- u u d u u u d d u u d u u u d u u u u d u u u
- u u u u u u u u d u u u u u u d d u u u d d s
- u d u d u u u u u d u d s s u u u d s u u u u
- d u u u d d u u u u u u d d s u u u d s u u d
- u u d u u u d d u u d u u u d s u u u d s s u
- u d u d u u u u u d u d u u u u u d u u u u u
- u u u u u u u u d u u u u u u d d u u u d d s
- u u u u u u u u d u u u u u u d d s u u d d s
Length of LCS: 17
LCS: AGCCTAAGGCTTAGCTT

LEETCODE



TASK B

```
X = input("Enter first string: ")
Y = input("Enter second string: ")

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

C = [[{'val': 0, 'dir': ''} for j in range(n+1)] for i 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] and (X != Y or i != j):
            C[i][j]['val'] = C[i-1][j-1]['val'] + 1
            C[i][j]['dir'] = 'd'
        else:
            if C[i-1][j]['val'] >= C[i][j-1]['val']:
                C[i][j]['val'] = C[i-1][j]['val']
                C[i][j]['dir'] = 'u'
            else:
                C[i][j]['val'] = C[i][j-1]['val']
                C[i][j]['dir'] = 's'

def backtrack(C, X, i, j):
    if i == 0 or j == 0:
        return ""
    if C[i][j]['dir'] == 'd':
        return backtrack(C, X, i-1, j-1) + X[i-1]
    elif C[i][j]['dir'] == 'u':
        return backtrack(C, X, i-1, j)
    else:
```

```
        return backtrack(C, X, i, j-1)

subseq = backtrack(C, X, m, n)

print("\nCost Matrix (values):")
for i in range(m+1):
    for j in range(n+1):
        print(C[i][j]['val'], end=" ")
    print()

print("\nDirection Matrix:")
for i in range(m+1):
    for j in range(n+1):
        print(C[i][j]['dir'] if C[i][j]['dir'] != '' else '-', end=" ")
    print()

if X == Y:
    print("\nLength of LRS:", C[m][n]['val'])
    print("LRS:", subseq)
else:
    print("\nLength of LCS:", C[m][n]['val'])
    print("LCS:", subseq)
```

```
Enter first string: AABEBCDD
Enter second string: AABEBCDD

Cost Matrix (values):
0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1
0 1 1 1 1 2 2 2 2
0 1 1 1 1 2 2 2 2
0 1 1 2 2 2 2 2 2
0 1 1 2 2 2 2 2 2
0 1 1 2 2 2 2 2 3
0 1 1 2 2 2 2 3 3

Direction Matrix:
- - - - - - - - -
- u d s s s s s s
- d u u u u u u u
- u u u u d s s s
- u u u u u u u u
- u u d s u u u u
- u u u u u u u u
- u u u u u u u d
- u u u u u u d u

Length of LRS: 3
LRS: ABD
```