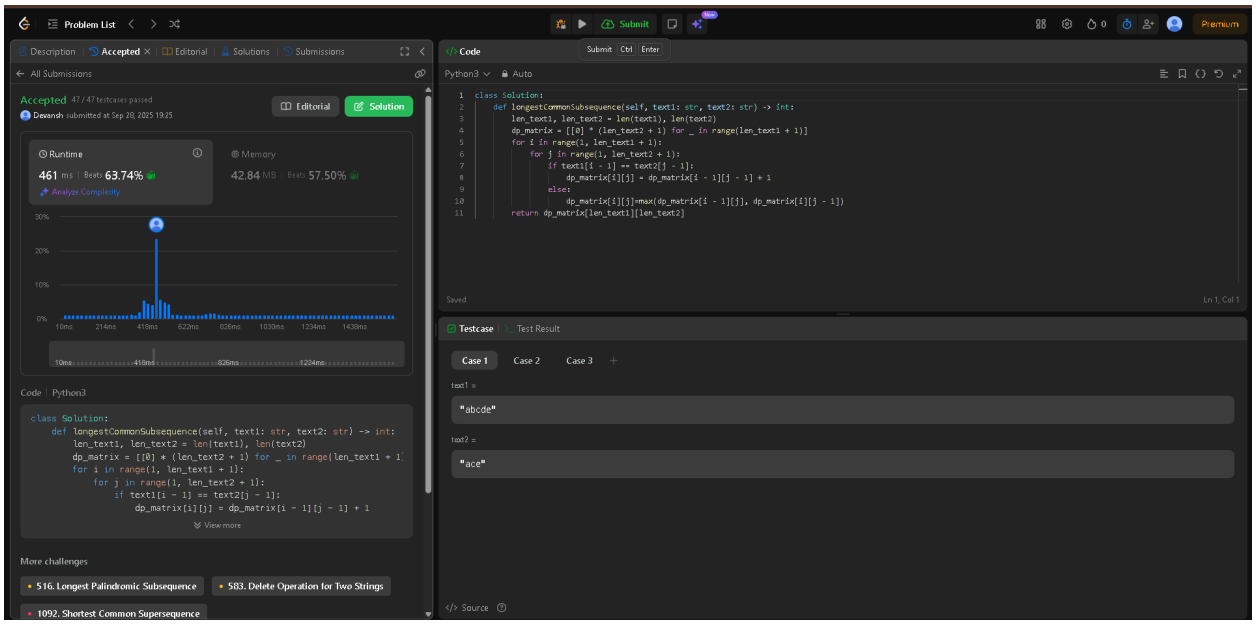
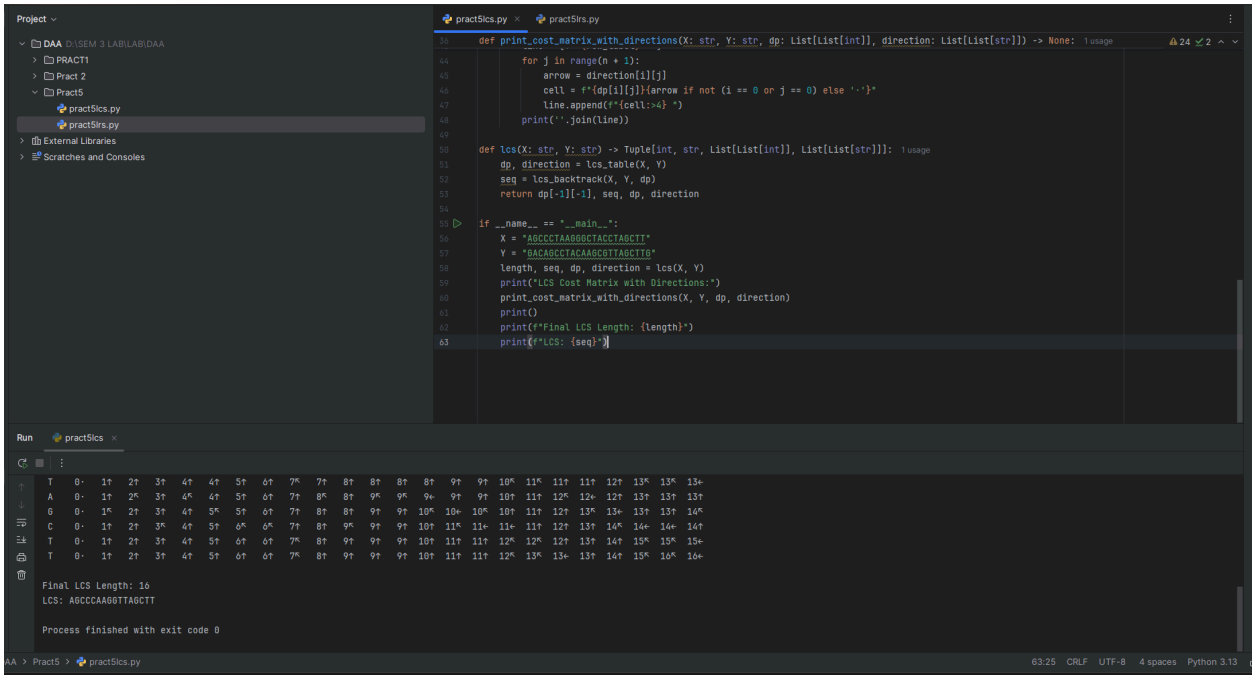
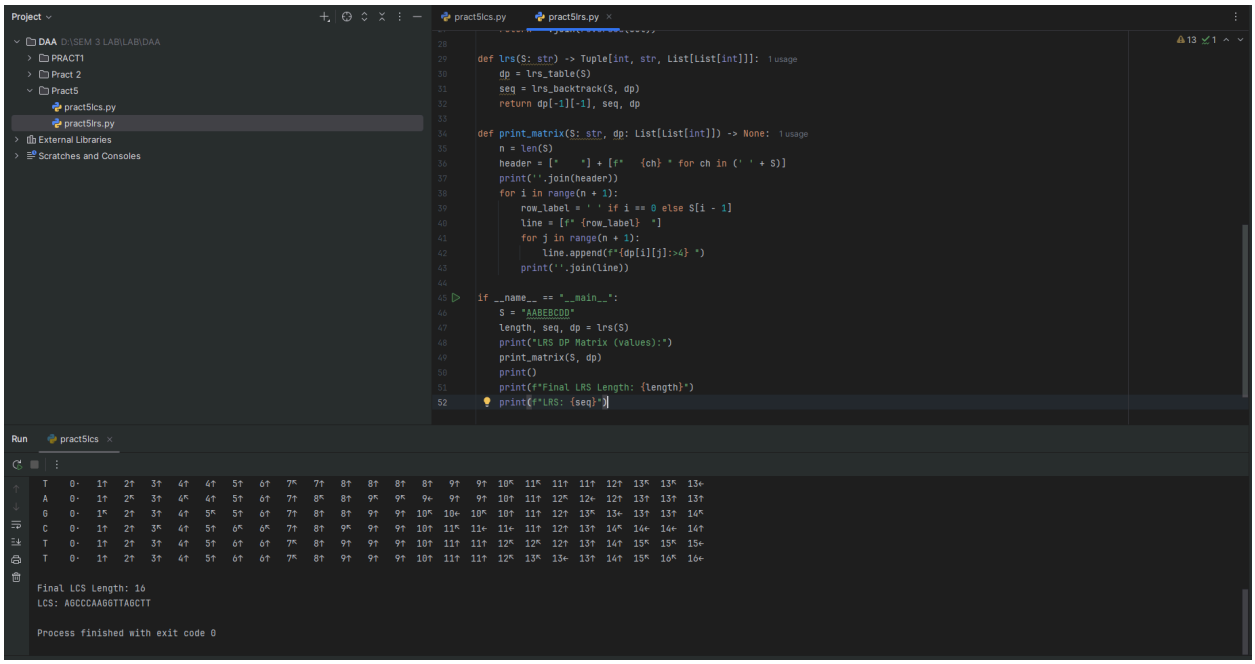

Problem List
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 47 / 47 testcases passed
Devansh submitted at Sep 28, 2025 19:25
Editorial
Solution
Runtime
461 ms | Beats 63.74%
Analyze Complexity
Memory
42.84 MB | Beats 57.50%
Code | Python3
class Solution:
def longestCommonSubsequence(self, text1: str, text2: str) -> int:
len_text1, len_text2 = len(text1), len(text2)
dp_matrix = [[0] * (len_text2 + 1) for _ in range(len_text1 + 1
for i in range(1, len_text1 + 1):
for j in range(1, len_text2 + 1):
if text1[i - 1] == text2[j - 1]:
dp_matrix[i][j] = dp_matrix[i - 1][j - 1] + 1
View more
More challenges
516. Longest Palindromic Subsequence
583. Delete Operation for Two Strings
1092. Shortest Common Supersequence
Code
Submit Ctrl Enter
Python3
Auto
class Solution:
def longestCommonSubsequence(self, text1: str, text2: str) -> int:
len_text1, len_text2 = len(text1), len(text2)
dp_matrix = [[0] * (len_text2 + 1) for _ in range(len_text1 + 1)]
for i in range(1, len_text1 + 1):
for j in range(1, len_text2 + 1):
if text1[i - 1] == text2[j - 1]:
dp_matrix[i][j] = dp_matrix[i - 1][j - 1] + 1
else:
dp_matrix[i][j]=max(dp_matrix[i - 1][j], dp_matrix[i][j - 1])
return dp_matrix[len_text1][len_text2]
Saved
Ln 1, Col 1
Testcase
Test Result
Case 1
Case 2
Case 3
text1 =
"abcde"
text2 =
"ace"
Source


Project
DAA D:\SEM 3 LAB\LAB\DAA
PRACT1
Pract 2
Pract5
pract5lcs.py
pract5lrs.py
External Libraries
Scratches and Consoles
pract5lcs.py
pract5lrs.py
def print_cost_matrix_with_directions(X: str, Y: str, dp: List[List[int]], direction: List[List[str]]) -> None: 1 usage
for j in range(n + 1):
arrow = direction[i][j]
cell = f"{dp[i][j]}{arrow if not (i == 0 or j == 0) else '·'}"
line.append(f"{cell:>4} ")
print(''.join(line))
def lcs(X: str, Y: str) -> Tuple[int, str, List[List[int]], List[List[str]]]: 1 usage
dp, direction = lcs_table(X, Y)
seq = lcs_backtrack(X, Y, dp)
return dp[-1][-1], seq, dp, direction
if __name__ == "__main__":
X = "AGCCCTAAGGGCTACCTAGCTT"
Y = "GACAGCCTACAAGCGTTAGCTTG"
length, seq, dp, direction = lcs(X, Y)
print("LCS Cost Matrix with Directions:")
print_cost_matrix_with_directions(X, Y, dp, direction)
print()
print(f"Final LCS Length: {length}")
print(f"LCS: {seq}")
Run
pract5lcs
Final LCS Length: 16
LCS: AGCCCAAGGTTAGCTT
Process finished with exit code 0
63:25 CRLF UTF-8 4 spaces Python 3.13


Project
DAA D:\SEM 3 LAB\LAB\DAA
PRACT1
Pract 2
Pract5
pract5lcs.py
pract5lrs.py
External Libraries
Scratches and Consoles
pract5lcs.py
pract5lrs.py
def lrs(S: str) -> Tuple[int, str, List[List[int]]]: 1 usage
dp = lrs_table(S)
seq = lrs_backtrack(S, dp)
return dp[-1][-1], seq, dp
def print_matrix(S: str, dp: List[List[int]]) -> None: 1 usage
n = len(S)
header = ['   '] + [f'   {ch}' for ch in (' ' + S)]
print(''.join(header))
for i in range(n + 1):
row_label = ' ' if i == 0 else S[i - 1]
line = [f' {row_label} ']
for j in range(n + 1):
line.append(f'{dp[i][j]:>4} ')
print(''.join(line))
if __name__ == "__main__":
S = "AABEBCDD"
length, seq, dp = lrs(S)
print("LRS DP Matrix (values):")
print_matrix(S, dp)
print()
print(f"Final LRS Length: {length}")
print(f"LRS: {seq}")
Run
pract5lcs
T 0· 1↑ 2↑ 3↑ 4↑ 4↑ 5↑ 6↑ 7↖ 7↑ 8↑ 8↑ 8↑ 8↑ 9↑ 9↑ 10↖ 11↖ 11↑ 11↑ 12↑ 13↖ 13↖ 13←
A 0· 1↑ 2↖ 3↑ 4↖ 4↑ 5↑ 6↑ 7↑ 8↖ 8↑ 9↖ 9↖ 9← 9↑ 9↑ 10↑ 11↑ 12↖ 12← 12↑ 13↑ 13↑ 13↑
G 0· 1↖ 2↑ 3↑ 4↑ 5↖ 5↑ 6↑ 7↑ 8↑ 8↑ 9↑ 9↑ 10↖ 10← 10↖ 10↑ 11↑ 12↑ 13↖ 13← 13↑ 13↑ 14↖
C 0· 1↑ 2↑ 3↖ 4↑ 5↑ 6↖ 6↖ 7↑ 8↑ 9↖ 9↑ 9↑ 10↑ 11↖ 11← 11← 11↑ 12↑ 13↑ 14↖ 14← 14← 14↑
T 0· 1↑ 2↑ 3↑ 4↑ 5↑ 6↑ 6↑ 7↖ 8↑ 9↑ 9↑ 9↑ 10↑ 11↑ 11↑ 12↖ 12↖ 12↑ 13↑ 14↑ 15↖ 15↖ 15←
T 0· 1↑ 2↑ 3↑ 4↑ 5↑ 6↑ 6↑ 7↖ 8↑ 9↑ 9↑ 9↑ 10↑ 11↑ 11↑ 12↖ 13↖ 13← 13↑ 14↑ 15↖ 16↖ 16←
Final LCS Length: 16
LCS: AGCCCAABGTTAGCTT
Process finished with exit code 0