Design and Analysis of Algorithm (CS 412)
Instructor: Dr. Ayesha Enayet
Date: ___________________
CS 6th

SIS ID:_______________ Name:________________

Instructions: Attempt all the questions. Use of device(s) is not allowed. Use a Blue/Black pen.

A. Given two strings s1 and s2, your task is to design a dynamic programming solution to compute the **minimum number of operations** required to convert string s1 into string s2.
You are allowed to use the following operations: [2]

- **Insert a character**
- **Delete a character**
- **Replace a character**

Input: Two strings s1 and s2, each of length ≤ 1000, containing only lowercase English letters.

Output: An integer representing the minimum edit distance between s1 and s2

Example:

Input:
s1 = "intention"
s2 = "execution"

Output:
5

Explanation:

- intention → inention (delete 't')
- inention → enention (replace 'i' with 'e')
- enention → exention (replace 'n' with 'x')
- exention → executon (replace 'n' with 'u')
- executon → execution (insert 'i')

Tasks:

1. Derive the recurrence relation for computing the minimum edit distance.
2. Use your proposed recurrence relation to fill the given table.
3. Identify the worst-case complexity of the solution.

	0	E	X	E	C	U	T	I	O	N
0	0	1	2	3	4	5	6	7	8	9
I	1	1	2	3	4	5	6	6	7	8
N	2	2	2	3	4	5	6	7	7	7
T	3	3	3	3	4	5	5	6	7	8
E	4	3	4	3	4	5	6	6	7	8
N	5	4	4	4	4	5	6	7	7	7
T	6	5	5	5	5	5	5	6	7	8
I	7	6	6	6	6	6	6	5	6	7
O	8	7	7	7	7	7	7	6	5	6
N	9	8	8	8	8	8	8	7	6	5

Formulate the dynamic programming recurrence for the given problem:

$$M[i,j] = \{\min(M[i-1,j-1], M[i-1,j], M[I,j-1])+1,\ s1[i] \neq s2[j]\ \}$$

Complexity: O(n.m)

Hint: If s1[i] == s2[j], the current subproblem requires no operation—just reuse the solution to the previous subproblem. Otherwise, compute dp[i][j] as one plus the minimum of the previously computed values: deleting a character, inserting one, or replacing one. This is a minimization problem over these three choices.

B. Write down the complexity of the following algorithms [2]

   a. Matrix-chain multiplication (dynamic programming): O(n3)
   b. 0/1 knapsack (dynamic programming) : O(n.c)
   c. Fractional knapsack (greedy algorithm): O(nlgn)
   d. Prim's Algorithm : O(ElgV)

C. Write an algorithm to solve the **Fractional Knapsack problem** using a **greedy approach**. [1]

$$\text{Assume } \frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \geq \frac{v_n}{w_n}$$

A ← [0, 0, ..., 0], V ← 0

for i from 1 to n:

   if W = 0:

      return (V, A)

   a ← min(w_i, W)

   V ← V + a * (v_i / w_i)

w_i ← w_i − a

A[i] ← A[i] + a

W ← W − a

return (V, A)

Reference : https://www.digitalocean.com/community/tutorials/fractional-knapsack-cpp