Aim: To find minimum edit distance for converting source string to target string. Get the number of operations needed for the same. Get steps of how to convert source string to target string.

Usage : python3 111708049_LabAssign2_code.py sourceString targetString
Assumptions :
Operations allowed with their costs - Insertion = 1, Deletion = 1, Substitution = 2

Input :
Source string = “intention”
Target string = “execution”

Output expected :
Minimum Edit distance = 8
Insertion: 1, Deletion: 1, Substitution: 3
Steps:
delete i
substitute n with e
substitute t with x
insert c
substitute n with u

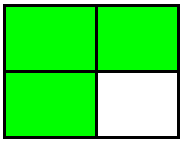
Algorithm:

Definition of local minima :

Min(
immediate upper cell + insertion-cost,
Immediate left cell + deletion-cost,
Immediate diagonal-upper cell + substitution-cost
)

I. Creating the minimum distance matrix :

Step 1 : Create a zero matrix with dimensions (len(target), len(source))
Step 2 : Fill the 0th row with values - 0, 1, 2, …, len(source)
Step 3 : Fill the 0th column with values - 0, 1, 2, …, len(target)
Step 4 : Initialize remaining rows in following way
For col := [1, len(source)] :
For row := [1, len(target)] :
if(source[col-1] == target[row-1]) :
Copy the previous diagonal entry
Else :
Fill the matrix position with local minima



II. Getting Steps to convert source string to target string(Backtracking) :

Starting from the lower-right corner of the completed matrix, repeat the following until you reach the upper-left corner: (row, col >= 0, 0)

1. Look at the neighboring cell to the upper left. If it doesn't exist, go to step 4. If the cell does exist, note the value stored there.
2. If the value in the upper-left cell equal to the value in the current cell, then
    - If **character at the current row position of target string** and **character at current column position of source string** are equal:
        - Record an empty operation (i.e. Equal). No edit was required in this case because the characters at this location are the same.
        - Update the current cell, moving diagonally up adding “diag” in ptr list
        - Return to step 1
    - Else find the minimum of immediate upward and leftward cells.
        - If the minimum is an immediate upward cell, go to step 5.
        - If the minimum is an immediate leftward cell, go to step 6.
3. Otherwise, do a three-way comparison between the cell to the left, the cell to the upper left, and the cell above. Pick the one with the smallest value. If there are multiple candidates, you can pick one at random;
    - If you picked the cell above, go to step 5.
    - If you picked the cell to the left, go to step 6.
    - If you picked the cell to the upper left(diagonal), go to step 7.
4. There are many branches here:
    - If there is no cell to the left and no cell above, then you are in the upper-left corner, and the algorithm has completed.
    - If there is no cell to the left, go to step 5. (This will continue in a loop until you reach the upper-left corner.)
    - If there is no cell above, go to step 6. (This will continue in a loop until you reach the upper-left corner.)
5. You are moving up. Do the following:
    - Increment insertion operation count, add “up” in ptr list.
    - Update the current cell, moving up.
    - Add this “insertion” step in steps list
    - Return to step 1.
6. You are moving left. Do the following:
    - Increment deletion operation count, add “left” in ptr list.
    - Update the current cell, moving left.
    - Add this “deletion” step in steps list
    - Return to step 1.
7. You are moving diagonally. Do the following:
    - Increment substitution operation count, add “diag” in ptr list.
    - Update the current cell, moving diagonally up.
    - Add this “substitution” step in steps list
    - Return to step 1.

Results: For “intention” to “execution”

Minimum edit distance matrix :

[[0 1 2 3 4 5 6 7 8 9]

 [1 2 3 4 3 4 5 6 7 8]

 [2 3 4 5 4 5 6 7 8 9]

 [3 4 5 6 5 6 7 8 9 10]

 [4 5 6 7 6 7 8 9 10 11]

 [5 6 7 8 7 8 9 10 11 12]

 [6 7 8 7 8 9 8 9 10 11]

 [7 6 7 8 9 10 9 8 9 10]

 [8 7 8 9 10 11 10 9 8 9]

 [9 8 7 8 9 10 11 10 9 8]]

Traversal while backtracking(indices) :

(9 9) (8 8) (7 7) (6 6) (5 5) (4 4) (3 4) (2 3) (1 2) (0 1) (0 0)

Ptr list :

['diag', 'diag', 'diag', 'diag', 'diag', 'up', 'diag', 'diag', 'diag', 'left']

Insertion : 1, Deletion : 1, Substitution : 3

Steps list : ['delete i', 'substitute n with e', 'substitute t with x', 'insert c', 'substitute n with u']

Conclusion:

- Implemented minimum edit distance algorithm using dynamic programming.
- Backtracked the steps for conversion.
- Stored the steps as well as directions while backtracking in the “steps” and “ptr” list respectively.