Data Analysis and Algorithm

Practical 5

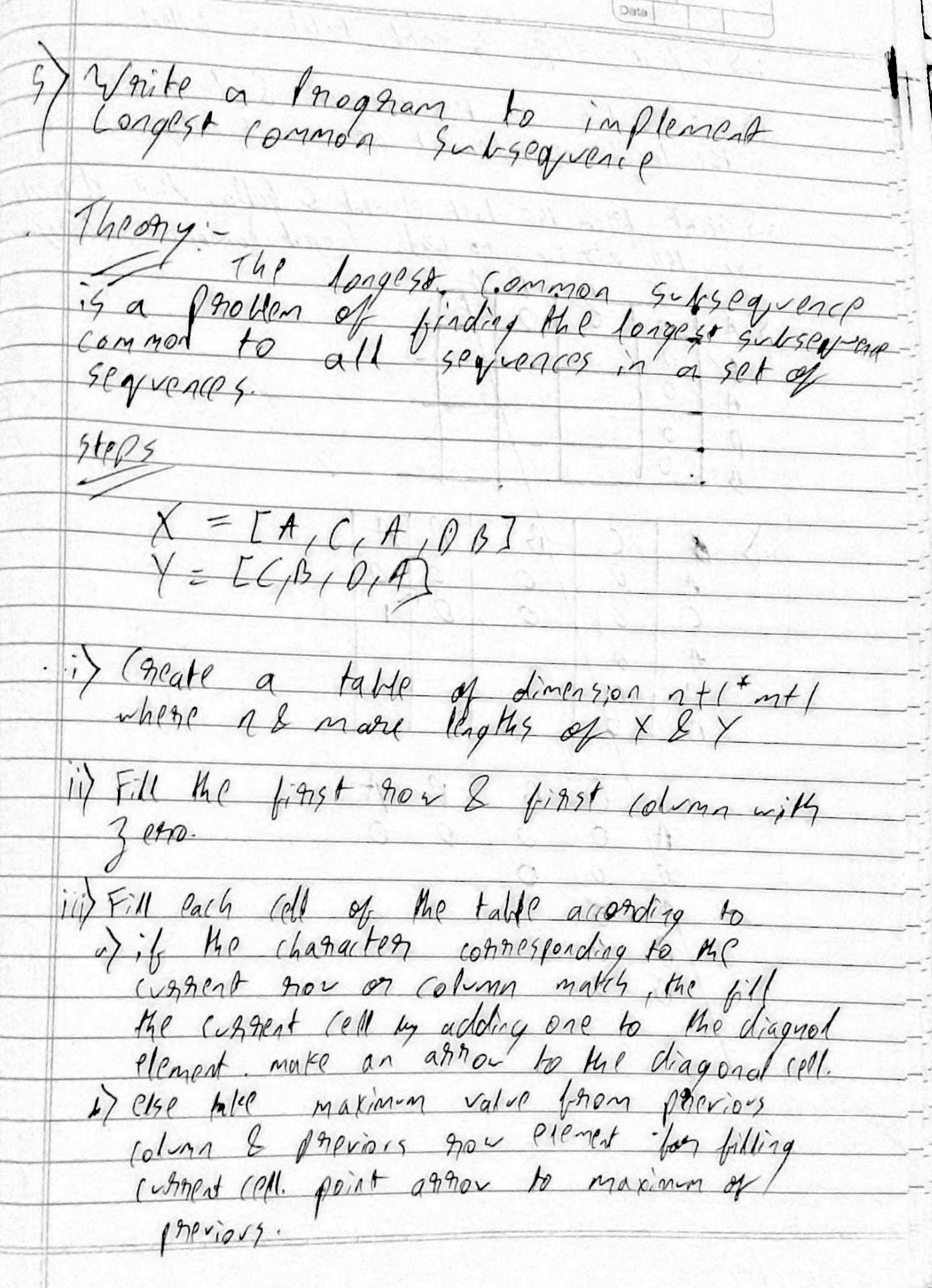
Write a program to find the longest common subsequence.

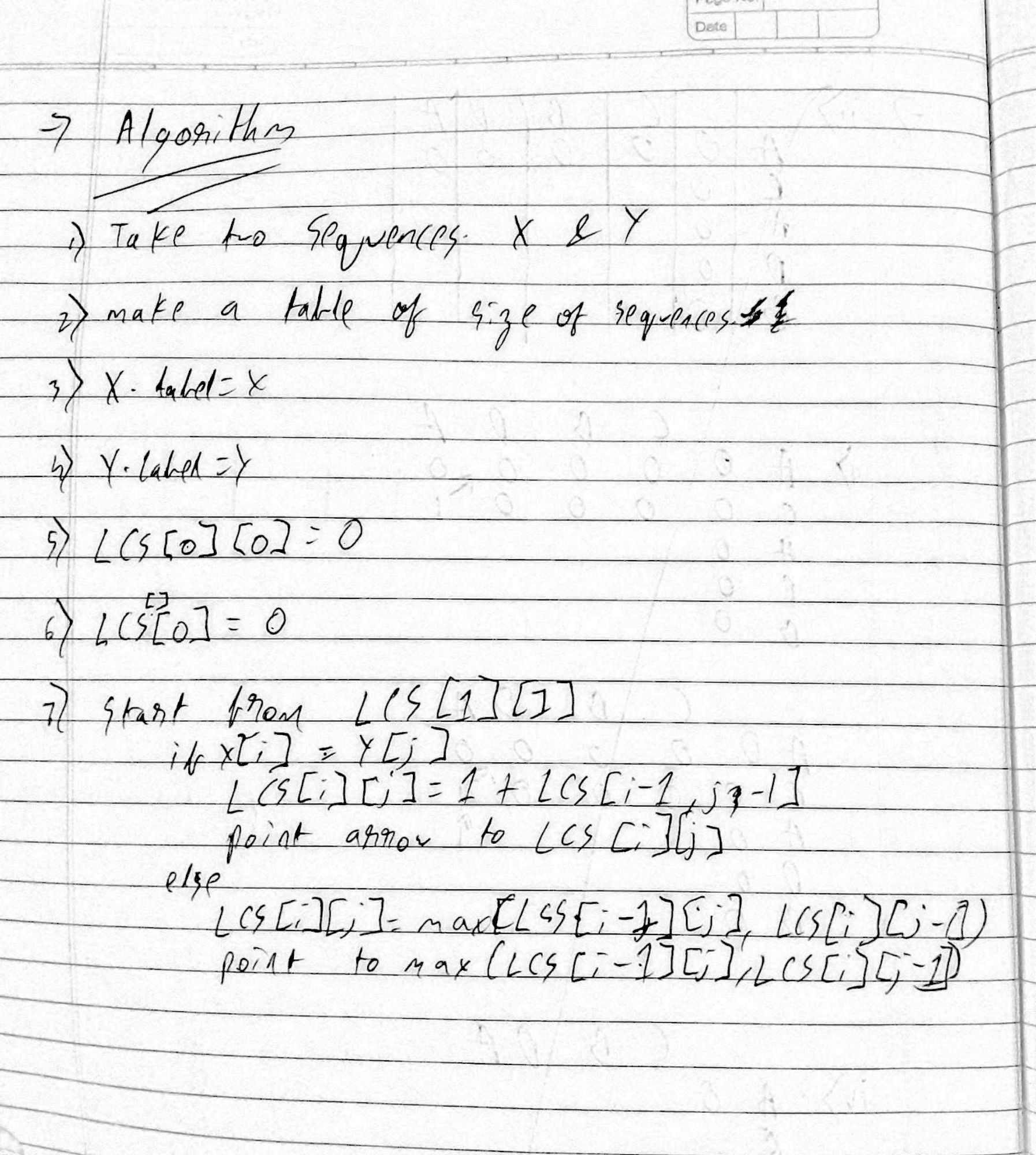
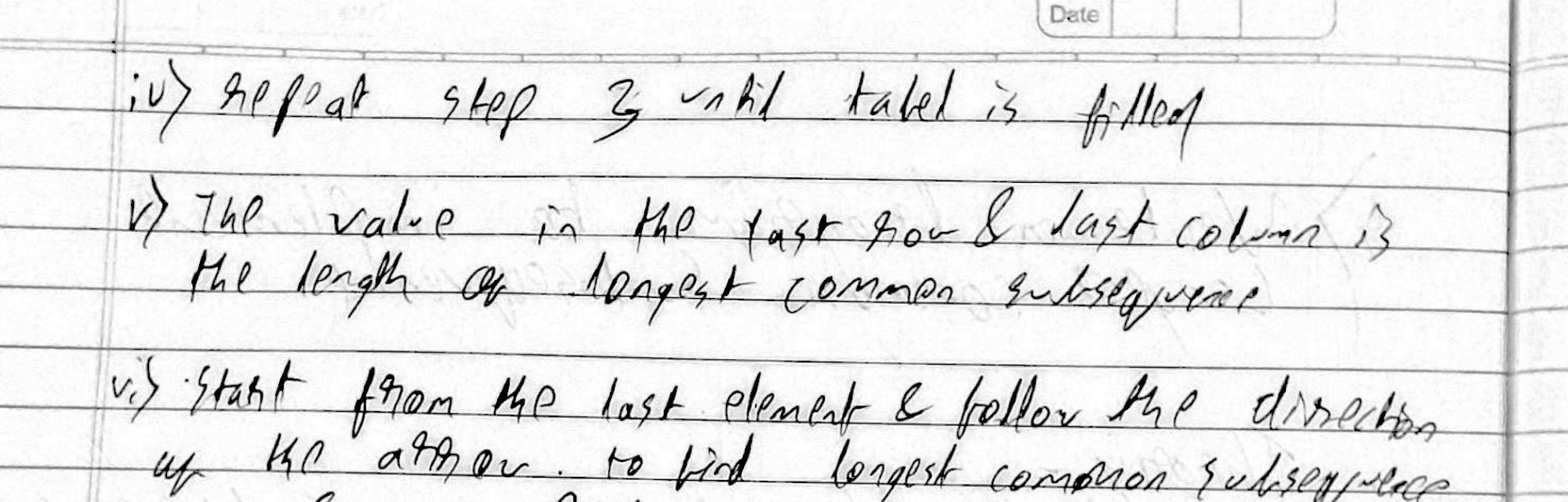
Date.: 24-09-21

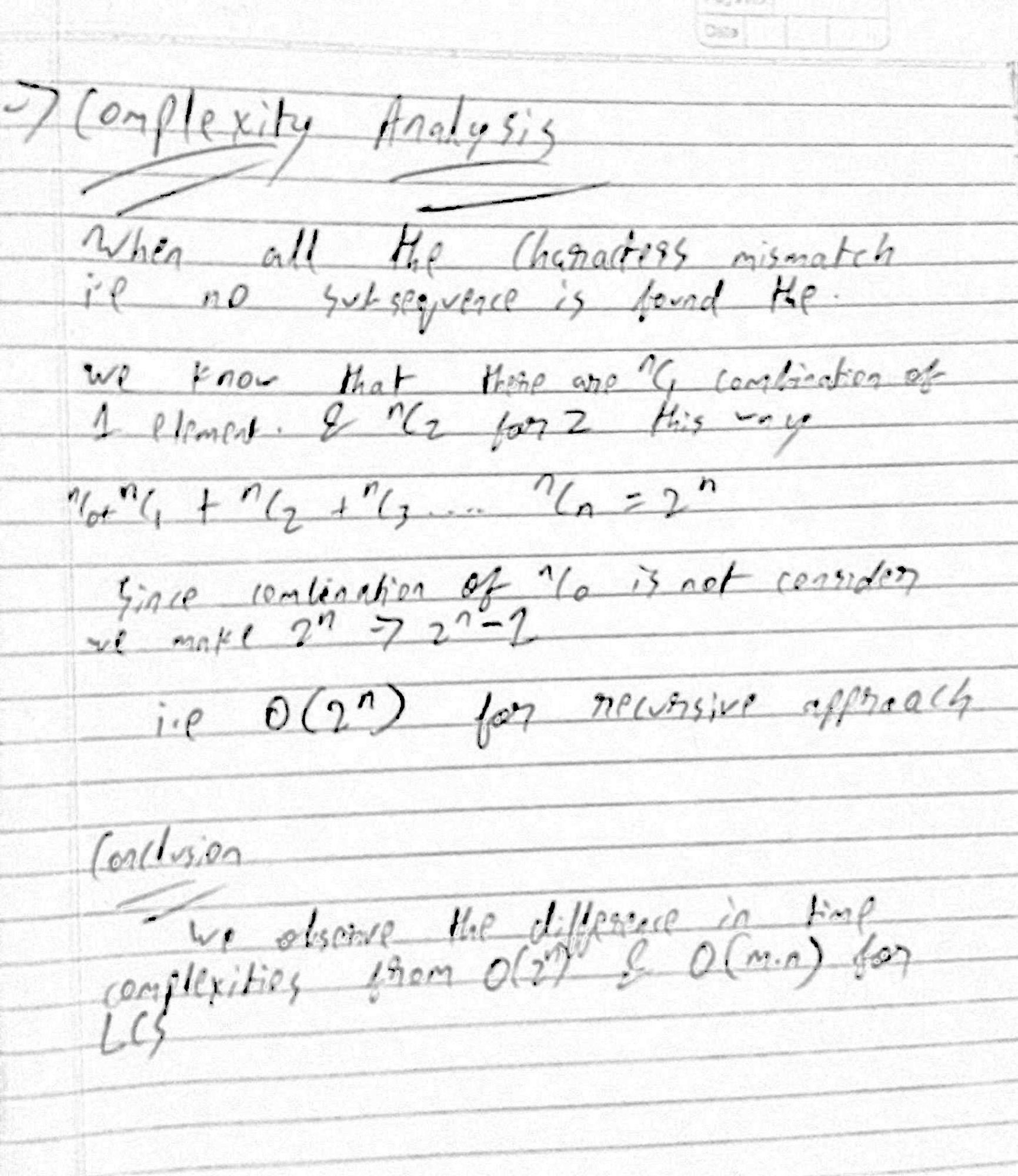
Name – Yash Vasudeo Prajapati

Rollno - 022

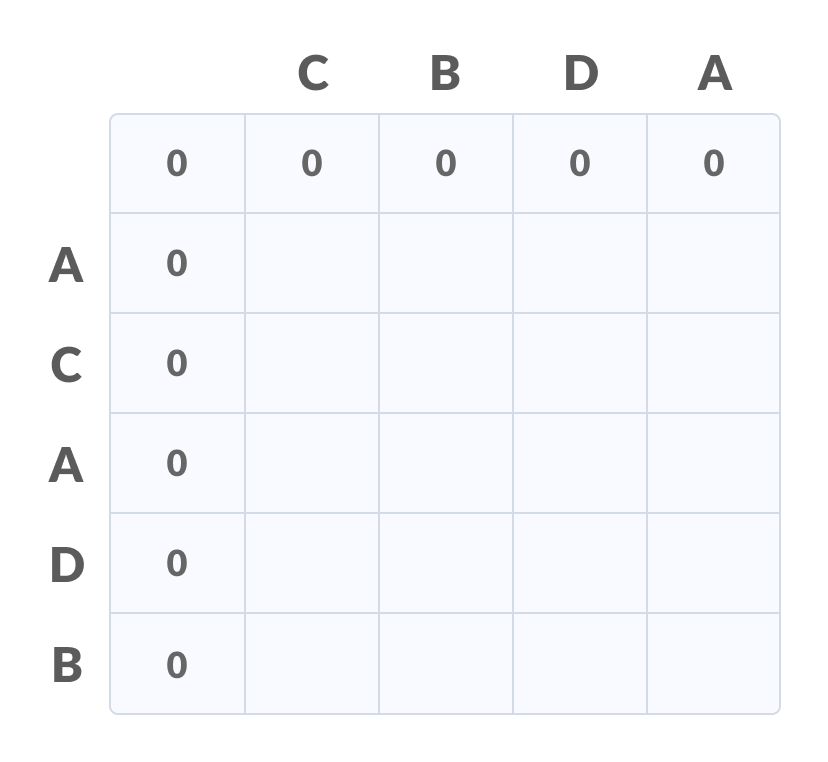
MSc. Computer Science

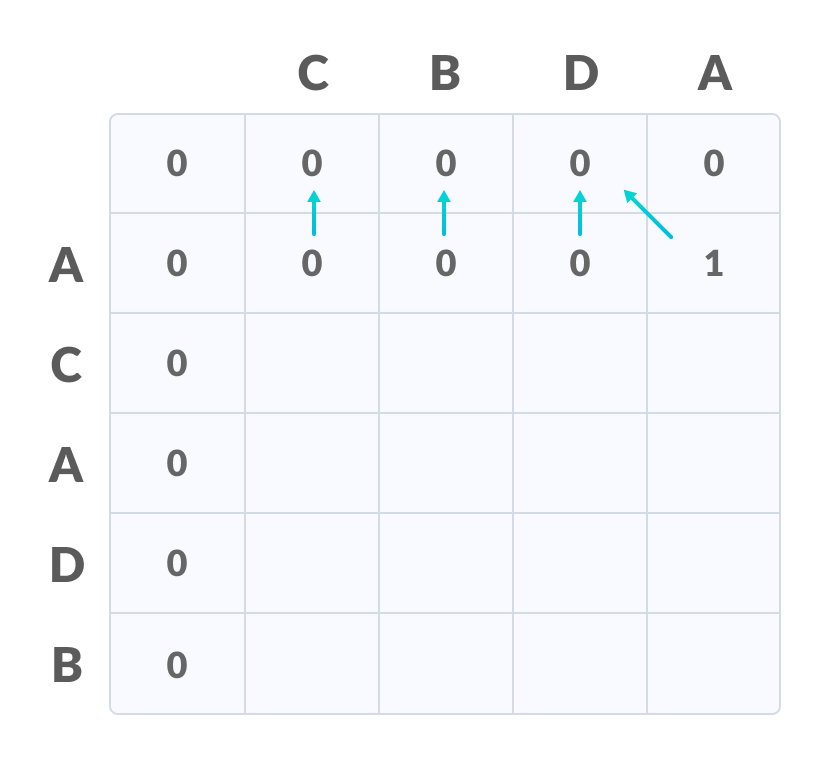


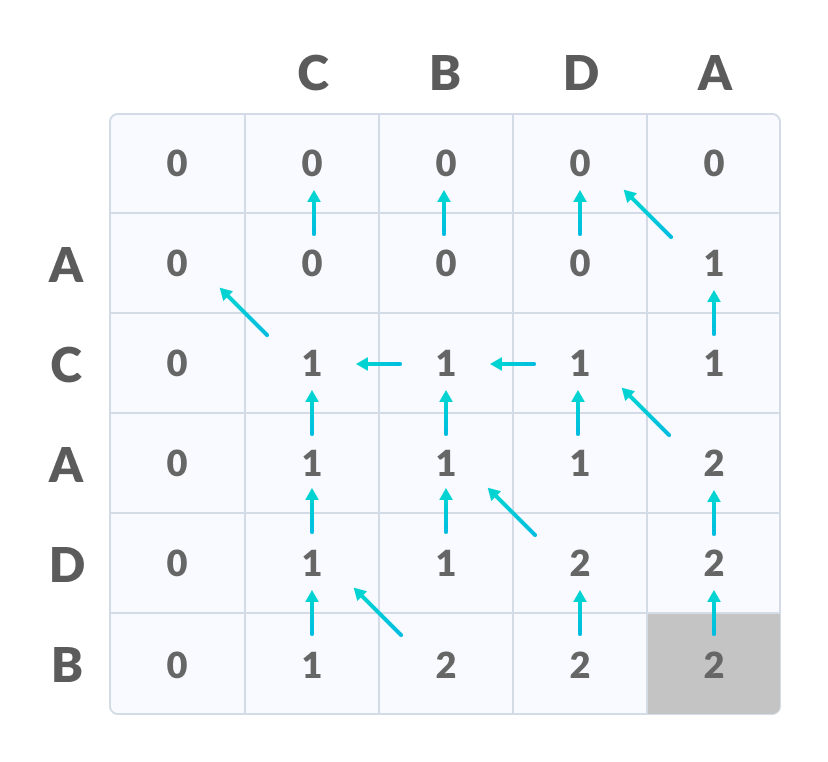
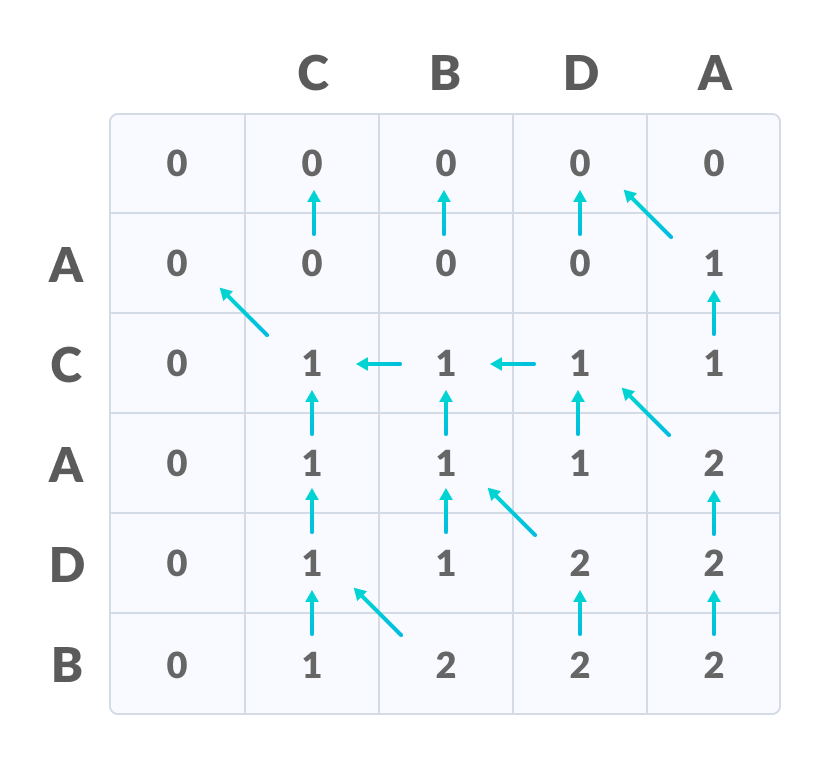




Example:-







Program

Recursive approach

def LCSLength(X, Y, m, n):

# return if the end of either sequence is reached

if m == 0 or n == 0:

return 0

# if the last character of `X` and `Y` matches

if X[m - 1] == Y[n - 1]:

return LCSLength(X, Y, m - 1, n - 1) + 1

# otherwise, if the last character of `X` and `Y` don't match

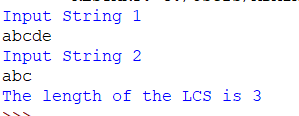
return max(LCSLength(X, Y, m, n - 1), LCSLength(X, Y, m - 1, n))

if \_\_name\_\_ == '\_\_main\_\_':

X = input("Input String 1 \n").upper()

Y = input("Input String 2 \n").upper()

print('The length of the LCS is', LCSLength(X, Y, len(X), len(Y)))



Complexity: - O(2^n)

Dynamic Approach

def LCSLength(X, Y, m, n, lookup={}):

    # return if the end of either string is reached

    if m == 0 or n == 0:

        return 0

    # construct a unique key from dynamic elements of the input

    key = (m, n)

    # if the subproblem is seen for the first time, solve it and

    # store its result in a dictionary

    if key not in lookup:

        # if the last character of `X` and `Y` matches

        if X[m - 1] == Y[n - 1]:

            lookup[key] = LCSLength(X, Y, m - 1, n - 1, lookup) + 1

        else:

            # otherwise, if the last character of `X` and `Y` don't match

            lookup[key] = max(LCSLength(X, Y, m, n - 1, lookup),

                            LCSLength(X, Y, m - 1, n, lookup))

    # return the subproblem solution from the dictionary

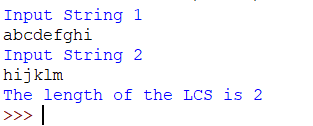
    return lookup[key]

if \_\_name\_\_ == '\_\_main\_\_':

X = input("Input String 1 \n").upper()

Y = input("Input String 2 \n").upper()

    print('The length of the LCS is', LCSLength(X, Y, len(X), len(Y)))



Complexity:- O(m+n)