CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 2 – Dynamic Programming

Neha Goud Baddam
11519516

Longest Common Sequence (LCS):

The **longest common subsequence** (**LCS**) **problem** is the problem of finding the longest common sequences in a set of sequences.

Definition:

1. Initially, the code will take the strings as input. Then it calculates the length of each string and prints it for the user to verify.

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

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

print("\nX value is ", X)
print("\nY value is ", Y)
```

2. Now, we call the function lcs(X, Y, m, n) by passing the two strings and their lengths to calculate the tables B and C. This function returns table B and table C. Next, we call the function compute_lcs(c, m, n) by passing table c and the lengths of the strings. This function computes the final output. Finally, we call the function print_lcs(b, c, m, n) to print the final output along with tables B and C.

```
b,c =  lcs(X,Y,m,n)

result = compute_lcs(c,m,n)

print_lcs(b,c,m,n)

print("\nLongest common sequence between X and Y is : ","".join(result))
```

LCS Algorithm:

i. Firstly, the function takes X and Y strings as input and their lengths m and n as input.

ii. Now, we populate tables B and C based on the below recursive formula.

Recursive formulation

Define $c[i, j]$ = length of LCS of X_i and Y_j. We want $c[m, n]$.

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0\,, \\ c[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j\,, \\ \max(c[i-1, j], c[i, j-1]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j\,. \end{cases}$$

iii. The function finally returns the final tables B and C.

CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 2 – Dynamic Programming

Neha Goud Baddam
11519516

```
# The longest common subsequence in Python
def lcs(X, Y, m, n):

    C = [[0 for x in range(n+1)] for x in range(m+1)]
    B = [[0 for x in range(n+1)] for x in range(m+1)]

    # Building the mtrix in bottom-up way
    for i in range(m+1):
        for j in range(n+1):
            if i == 0 or j == 0:
                C[i][j] = 0
            elif X[i-1] == Y[j-1]:
                C[i][j] = C[i-1][j-1] + 1
                B[i][j] = u'\u2b09'

            else:

                if C[i-1][j] > C[i][j-i]:
                    B[i][j] = u'\u2b06'
                else:
                    B[i][j] =  u'\u2190'

                C[i][j] = max(C[i-1][j], C[i][j-1])
    return B,C
```

iv. The function compute_lcs(C, m, n) uses the computed C table to get the longest common sequence from the table using the optimized algorithm as shown below. This function returns the result, the longest common sequence of the two input strings X and Y.

```
def compute_lcs(C,m,n):

    i = m
    j = n

    x = C[i][j]

    res = [""] * (x+1)

    while i > 0 and j > 0:
        if X[i-1] == Y[j-1]:
            res[x-1] = X[i-1]
            i -= 1
            j -= 1
            x -= 1
        elif C[i-1][j] > C[i][j-1]:
            i -= 1
        else:
            j -= 1

    return res
```

v. Finally, we print all the results and tables using the function print_lcs(b, c, m, n).

CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 2 – Dynamic Programming

Neha Goud Baddam
11519516

```
def print_lcs(b,c,m,n):

    for i in range(1,n+1):
        b[0][i] = Y[i-1]
        i=i+1
    for i in range(1,m+1):
        b[i][0] = X[i-1]
        i=i+1

    print("\nTable B:")
    for row in b:
        for col in row:
            print(col,end = " ")
        print()

    print("\nTable C: ")
    for row in c:
        for col in row:
            print(col,end = " ")
        print()
```

Code for LCS:

CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 2 – Dynamic Programming

Neha Goud Baddam
11519516

In [60]:
```
# The longest common subsequence in Python
def lcs(X, Y, m, n):

    C = [[0 for x in range(n+1)] for x in range(m+1)]
    B = [[0 for x in range(n+1)] for x in range(m+1)]

    # Building the mtrix in bottom-up way
    for i in range(m+1):
        for j in range(n+1):
            if i == 0 or j == 0:
                C[i][j] = 0
            elif X[i-1] == Y[j-1]:
                C[i][j] = C[i-1][j-1] + 1
                B[i][j] = u'\u2b09'

            else:

                if C[i-1][j] > C[i][j-i]:
                    B[i][j] = u'\u2b06'
                else:
                    B[i][j] =  u'\u2190'

                C[i][j] = max(C[i-1][j], C[i][j-1])
    return B,C
```

```
def compute_lcs(C,m,n):

    i = m
    j = n

    x = C[i][j]

    res = [""] * (x+1)

    while i > 0 and j > 0:
        if X[i-1] == Y[j-1]:
            res[x-1] = X[i-1]
            i -= 1
            j -= 1
            x -= 1
        elif C[i-1][j] > C[i][j-1]:
            i -= 1
        else:
            j -= 1

    return res
```

```
def print_lcs(b,c,m,n):

    for i in range(1,n+1):
        b[0][i] = Y[i-1]
        i=i+1
    for i in range(1,m+1):
        b[i][0] = X[i-1]
        i=i+1

    print("\nTable B:")
    for row in b:
        for col in row:
            print(col,end = " ")
        print()

    print("\nTable C: ")
    for row in c:
        for col in row:
            print(col,end = " ")
        print()
```

CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 2 – Dynamic Programming

Neha Goud Baddam
11519516

In [61]:
```
X = str(input("Enter first string "))
Y = str(input("Enter second string "))

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

print("\nX value is ", X)
print("\nY value is ", Y)

b,c =  lcs(X,Y,m,n)

result = compute_lcs(c,m,n)

print_lcs(b,c,m,n)

print("\nLongest common sequence between X and Y is : ","".join(result))
```

Execution/ Output:

1. **For given inputs “spanking” and “amputation”**

CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 2 – Dynamic Programming

Neha Goud Baddam
11519516

```
Enter first string spanking
Enter second string amputation

X value is  spanking

Y value is  amputation

Table B:
0 a m p u t a t i o n
s ← ← ← ← ← ← ← ← ← ←
p ← ← ↖ ← ← ← ← ← ← ←
a ↖ ← ↑ ← ← ↖ ← ← ← ←
n ↑ ↑ ↑ ↑ ← ↑ ↑ ↑ ↑ ↖
k ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
i ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↖ ↑ ↑
n ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↖
g ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

Table C:
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
0 1 1 1 1 1 2 2 2 2 2
0 1 1 1 1 1 2 2 2 2 3
0 1 1 1 1 1 2 2 2 2 3
0 1 1 1 1 1 2 2 3 3 3
0 1 1 1 1 1 2 2 3 3 4
0 1 1 1 1 1 2 2 3 3 4

Longest common sequence between X and Y is :  pain
```

2. **<u>For other input AGGTAB and GXTXAYB</u>**

CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 2 – Dynamic Programming

Neha Goud Baddam
11519516

```
Enter first string AGGTAB
Enter second string GXTXAYB

X value is  AGGTAB

Y value is  GXTXAYB

Table B:
0 G X T X A Y B
A ← ← ← ← ↖ ← ←
G ↖ ← ← ← ← ← ←
G ↖ ↑ ↑ ← ← ← ←
T ↑ ↑ ↖ ↑ ← ← ←
A ↑ ↑ ↑ ↑ ↖ ↑ ↑
B ↑ ↑ ↑ ↑ ↑ ↑ ↖

Table C:
0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1
0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1
0 1 1 2 2 2 2 2
0 1 1 2 2 3 3 3
0 1 1 2 2 3 3 4

Longest common sequence between X and Y is :  GTAB
```

3. **For other input ABCDGH and AEDFHR**

CSCE 5150 – Analysis of Computer Algorithms

Programming Assignment 2 – Dynamic Programming

Neha Goud Baddam
11519516

```
Enter first string ABCDGH
Enter second string AEDFHR

X value is  ABCDGH

Y value is  AEDFHR

Table B:
0 A E D F H R
A ↖ ← ← ← ← ←
B ↑ ↑ ← ← ← ←
C ↑ ↑ ↑ ← ← ←
D ↑ ↑ ↖ ↑ ← ←
G ↑ ↑ ↑ ↑ ↑ ↑
H ↑ ↑ ↑ ↑ ↖ ↑

Table C:
0 0 0 0 0 0 0
0 1 1 1 1 1 1
0 1 1 1 1 1 1
0 1 1 1 1 1 1
0 1 1 2 2 2 2
0 1 1 2 2 2 2
0 1 1 2 2 3 3

Longest common sequence between X and Y is :  ADH
```