Hongqi Guo

```
LCS(x, y)
1.  a ← length[x]
    b ← length[y]
    for x ← 0 to a do
        for y ← 0 to b do.
            if ( x = 0 or y = 0 )
                lengthLCS[i, j] ← 0
            else if a[i-1] = b[j-1]
                LengthLCS[i, j] ← LengthLCS[i-1, j-1] + 1.
            else
            LengthLCS[i, j] ← max(lengthLCS[i-1, j], LengthLCS[i, j-1])
        pointer ← lengthLCS[a, b]
        i ← a
        j ← b.
        while i > 0 and j > 0.
        do if a[i-1] = b[j-1]
            seq[pointer - 1] = x[i-1]
```

```
        i ← i-1
        j ← j-1.
        pointer ← pointer -1.
    else if LengthLCS [i-1, j] > lengthLCS [i, j-1]
        i ← i-1
    else. j ← j-1

    z ← ' '

for u ← 0 to Length[seq] -3 do
    if seq[u] = x and seq[u+1] = y and seq[u+2] = y
        u -> u+2
    else
        z = z + seq[u]
    return z.

Space complexity = O(n)
Time complexity = O(n·n)
```

2. We generate all the subsequences of a sequence, then we compare each subsequences with other sequences to check if they are common. and then find the subsequences of maximum length.

then we have a string of length n. subsequences of length 1, 2, 3 -- n, so we can use permutation and combination.

sub-sequences = $^nC_1 + {}^nC_2 + \cdots {}^nC_n$.

So a string of length n has $2^n - 1$ different possible subsequences since we don't consider the subsequence with length 0.

to check if a subsequence is common to both the string or not.

then $O(n)$ time.

so total time needed to check all the sub-sequences $= n \cdot 2^n - 1$.

time complexity be brute force method is $O(n \cdot 2^n - 1)$