Printing Items in 0/1 Knapsack & Unbounded Knapsack
Temple Offerings & Egg Dropping Puzzle
Dice Throw & Word Break Problem
Vertex Cover Problem & Tile Stacking Problem
Box Stacking Problem
Highway Billboard Problem & Largest Independent Set Problem
Print equal sum sets of array (Partition problem) | Set 1 & Set 2
Longest Bitonic Subsequence
Printing Longest Bitonic Subsequence
Print Longest Palindromic Subsequence
Longest palindrome subsequence with O(n) space
Explore more...
Hard Problems
Palindrome Partitioning & Word Wrap Problem
Mobile Numeric Keypad Problem & The painter's partition problem
Boolean Parenthesization Problem & Bridge and Torch problem
A Space Optimized DP solution for 0-1 Knapsack Problem
Matrix Chain Multiplication & Printing brackets in Matrix Chain Multiplication Problem
Number of palindromic paths in a matrix
Largest rectangular sub-matrix whose sum is 0
Largest rectangular sub-matrix having sum divisible by k
Maximum sum bitonic subarray
K maximum sums of overlapping contiguous sub-arrays
Maximum profit by buying and selling a share at most k times
Maximum points from top left of matrix to bottom right and return back
Check whether row or column swaps produce maximum size binary sub-matrix with all 1s
Minimum number of elements which are not part of Increasing or decreasing subsequence in array
Count ways to increase LCS length of two strings by one
Count of AP (Arithmetic Progression) Subsequences in an array
Explore more...

Longest Common Substring | DP-29

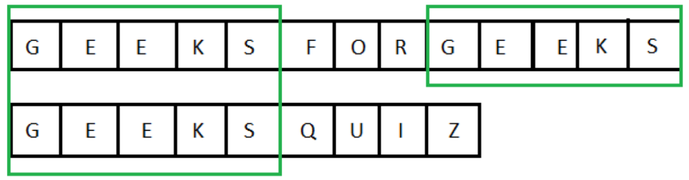
Given two strings ‘X’ and ‘Y’, find the length of the longest common substring.

Examples :

```
Input : X = "GeeksforGeeks", y = "GeeksQuiz"
Output : 5
The longest common substring is "Geeks" and is of
length 5.

Input : X = "abcdxyz", y = "xyzabcd"
Output : 4
The longest common substring is "abcd" and is of
length 4.

Input : X = "zxabcdezy", y = "yzabcdezx"
Output : 6
The longest common substring is "abcdez" and is of
length 6.
```



OUTPUT: 5
As longest Common String is "Geeks"

Recommended: Please solve it on "*PRACTICE* " first, before moving on to the solution.

Let m and n be the lengths of first and second strings respectively.

A **simple solution** is to one by one consider all substrings of first string and for every substring check if it is a substring in second string. Keep track of the maximum length substring. There will be O(m^2) substrings and we can find whether a string is subsring on another string in O(n) time (See this). So overall time complexity of this method would be O(n * m^2)

Dynamic Programming can be used to find the longest common substring in O(m*n) time. The idea is to find length of the longest common suffix for all substrings of both strings and store these lengths in a table.

```
The longest common suffix has following optimal substructure property
   LCSuff(X, Y, m, n) = LCSuff(X, Y, m-1, n-1) + 1 if X[m-1] = Y[n-1]
                        0  Otherwise (if X[m-1] != Y[n-1])

The maximum length Longest Common Suffix is the longest common substring.
   LCSubStr(X, Y, m, n)  = Max(LCSuff(X, Y, i, j)) where 1 <= i <= m
                                                 and 1 <= j <= n
```

Following is the implementation of the above solution.

C++

```
/* Dynamic Programming solution to find length of the
   longest common substring */
#include<iostream>
#include<string.h>
using namespace std;

// A utility function to find maximum of two integers
int max(int a, int b)
{   return (a > b)? a : b; }

/* Returns length of longest common substring of X[0..m-1]
   and Y[0..n-1] */
int LCSubStr(char *X, char *Y, int m, int n)
{
    // Create a table to store lengths of longest common suffixes of
    // substrings.   Notethat LCSuff[i][j] contains length of longest
    // common suffix of X[0..i-1] and Y[0..j-1]. The first row and
    // first column entries have no logical meaning, they are used only
    // for simplicity of program
    int LCSuff[m+1][n+1];
    int result = 0;  // To store length of the longest common substring

    /* Following steps build LCSuff[m+1][n+1] in bottom up fashion. */
    for (int i=0; i<=m; i++)
    {
        for (int j=0; j<=n; j++)
        {
            if (i == 0 || j == 0)
                LCSuff[i][j] = 0;

            else if (X[i-1] == Y[j-1])
            {
                LCSuff[i][j] = LCSuff[i-1][j-1] + 1;
                result = max(result, LCSuff[i][j]);
            }
            else LCSuff[i][j] = 0;
        }
    }
    return result;
}

/* Driver program to test above function */
int main()
{
    char X[] = "OldSite:GeeksforGeeks.org";
    char Y[] = "NewSite:GeeksQuiz.com";

    int m = strlen(X);
    int n = strlen(Y);

    cout << "Length of Longest Common Substring is "
         << LCSubStr(X, Y, m, n);
    return 0;
}
```

Run on IDE

Java

```java
//  Java implementation of finding length of longest
// Common substring using Dynamic Programming
public class LongestCommonSubSequence
{
    /*
       Returns length of longest common substring
       of X[0..m-1] and Y[0..n-1]
    */
    static int LCSubStr(char X[], char Y[], int m, int n)
    {
        // Create a table to store lengths of longest common suffixes of
        // substrings. Note that LCSuff[i][j] contains length of longest
        // common suffix of X[0..i-1] and Y[0..j-1]. The first row and
        // first column entries have no logical meaning, they are used only
        // for simplicity of program
        int LCStuff[][] = new int[m + 1][n + 1];
        int result = 0;  // To store length of the longest common substring

        // Following steps build LCSuff[m+1][n+1] in bottom up fashion
        for (int i = 0; i <= m; i++)
        {
            for (int j = 0; j <= n; j++)
            {
                if (i == 0 || j == 0)
                    LCStuff[i][j] = 0;
                else if (X[i - 1] == Y[j - 1])
                {
                    LCStuff[i][j] = LCStuff[i - 1][j - 1] + 1;
                    result = Integer.max(result, LCStuff[i][j]);
                }
                else
                    LCStuff[i][j] = 0;
            }
        }
        return result;
    }

    // Driver Program to test above function
    public static void main(String[] args)
    {
        String X = "OldSite:GeeksforGeeks.org";
        String Y = "NewSite:GeeksQuiz.com";

        int m = X.length();
        int n = Y.length();

        System.out.println("Length of Longest Common Substring is "
                + LCSubStr(X.toCharArray(), Y.toCharArray(), m, n));
    }
}

// This code is contributed by Sumit Ghosh
```

Run on IDE

Python3

```python
# Python3 implementation of Finding
# Length of Longest Common Substring

# Returns length of longest common
# substring of X[0..m-1] and Y[0..n-1]
def LCSubStr(X, Y, m, n):

    # Create a table to store lengths of
    # longest common suffixes of substrings.
    # Note that LCSuff[i][j] contains the
    # length of longest common suffix of
    # X[0...i-1] and Y[0...j-1]. The first
    # row and first column entries have no
    # logical meaning, they are used only
    # for simplicity of the program.

    # LCSuff is the table with zero
    # value initially in each cell
    LCSuff = [[0 for k in range(n+1)] for l in range(m+1)]

    # To store the length of
    # longest common substring
    result = 0

    # Following steps to build
    # LCSuff[m+1][n+1] in bottom up fashion
    for i in range(m + 1):
        for j in range(n + 1):
            if (i == 0 or j == 0):
                LCSuff[i][j] = 0
            elif (X[i-1] == Y[j-1]):
                LCSuff[i][j] = LCSuff[i-1][j-1] + 1
                result = max(result, LCSuff[i][j])
            else:
                LCSuff[i][j] = 0
    return result

# Driver Program to test above function
X = 'OldSite:GeeksforGeeks.org'
Y = 'NewSite:GeeksQuiz.com'

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

```
print('Length of Longest Common Substring is',
                    LCSubStr(X, Y, m, n))

# This code is contributed by Soumen Ghosh
```

Run on IDE

C#

```
// C# implementation of finding length of longest
// Common substring using Dynamic Programming
using System;

class GFG {

    // Returns length of longest common
    // substring of X[0..m-1] and Y[0..n-1]
    static int LCSubStr(string X, string Y,
                                int m, int n)
    {

        // Create a table to store lengths of
        // longest common suffixes of substrings.
        // Note that LCSuff[i][j] contains length
        // of longest common suffix of X[0..i-1]
        // and Y[0..j-1]. The first row and first
        // column entries have no logical meaning,
        // they are used only for simplicity of
        // program
        int[, ] LCStuff = new int[m + 1, n + 1];

        // To store length of the longest common
        // substring
        int result = 0;

        // Following steps build LCSuff[m+1][n+1]
        // in bottom up fashion
        for (int i = 0; i <= m; i++) {
            for (int j = 0; j <= n; j++) {
                if (i == 0 || j == 0)
                    LCStuff[i, j] = 0;
                else if (X[i - 1] == Y[j - 1]) {
                    LCStuff[i, j] =
                            LCStuff[i - 1, j - 1] + 1;

                    result = Math.Max(result,
                                      LCStuff[i, j]);
                }
                else
                    LCStuff[i, j] = 0;
            }
        }
        return result;
    }

    // Driver Program to test above function
    public static void Main()
    {
        String X = "OldSite:GeeksforGeeks.org";
        String Y = "NewSite:GeeksQuiz.com";

        int m = X.Length;
        int n = Y.Length;

        Console.Write("Length of Longest Common"
        + " Substring is " + LCSubStr(X, Y, m, n));
    }

}

// This code is contributed by Sam007.
```

Run on IDE

Output:

```
Length of Longest Common Substring is 10
```

Time Complexity: O(m*n)
Auxiliary Space: O(m*n)

References: http://en.wikipedia.org/wiki/Longest_common_substring_problem

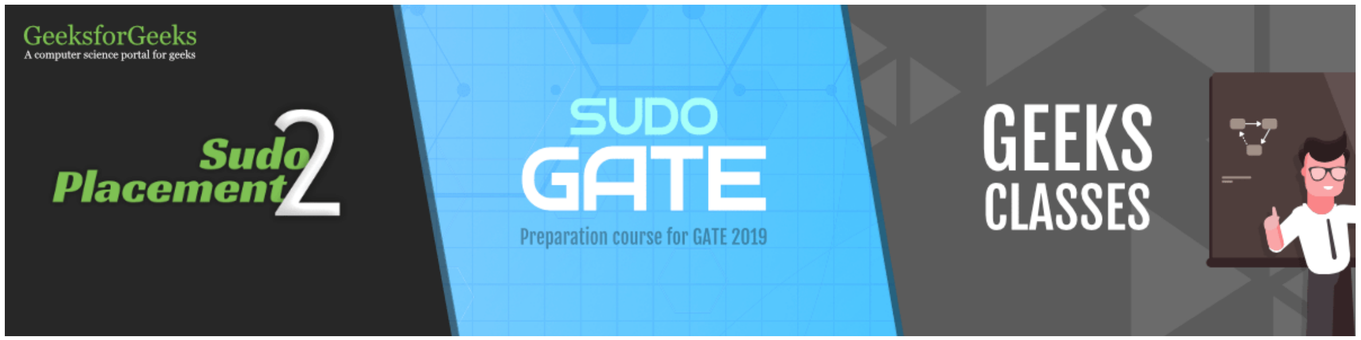
The longest substring can also be solved in O(n+m) time using Suffix Tree. We will be covering Suffix Tree based solution in a separate post.

Exercise: The above solution prints only length of the longest common substring. Extend the solution to print the substring also.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Practice Tags : Microsoft Strings Dynamic Programming

Article Tags : Dynamic Programming Strings Microsoft



Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Count all possible paths from top left to bottom right of a mXn matrix
Longest Common Subsequence | DP-4
Minimum insertions to form a palindrome | DP-28
Longest Palindromic Substring | Set 1
Program for nth Catalan Number
Alternate Fibonacci Numbers
Find the largest area rectangular sub-matrix whose sum is equal to k
Sum of elements of all partitions of number such that no element is less than K
Water Jug Problem using Memoization
Number of Co-prime pairs obtained from the sum of digits of elements in the given range

Logged in as **Dragan Nikolic**(Logout)

2.8 Average Difficulty : **2.8/5.0**
Based on **131** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

Give Feedback | Add your Notes | Improve Article

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments | Share this post!