Dynamic Programming | Set 29 (Longest Common Substring)

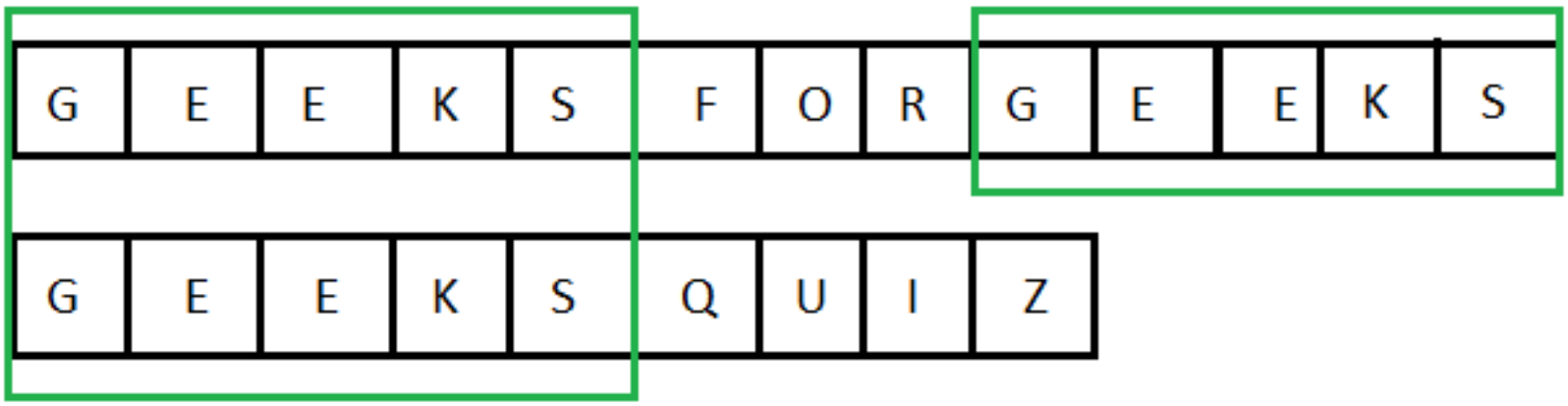
Given two strings 'X' and 'Y', find the length of the longest common substring.

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"

We strongly recommend that you click here and practice it, 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 sub-string 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 C++ implementation of the above solution.

```
/* 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

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

GATE CS Corner Company Wise Coding Practice

Dynamic Programming

Recommended Posts:

Dynamic Programming | Set 4 (Longest Common Subsequence)
Dynamic Programming | Set 30 (Dice Throw)
Dynamic Programming | Set 28 (Minimum insertions to form a palindrome)
Dynamic Programming | Set 12 (Longest Palindromic Subsequence)
Dynamic Programming | Set 3 (Longest Increasing Subsequence)

(Login to Rate and Mark)

2.5 Average Difficulty : **2.5/5.0**
Based on **35** vote(s)

Add to TODO List
Mark as DONE

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

Share this post!

{# The script below injects the embed bootloader and then modifies CSP rules to disallow inline scripts while keeping the rest of the rules defined automatically by headers intact with a very broad `https:` rule. Keep in mind that this still forces the page to load all scripts via HTTPS. #}

74 Comments GeeksforGeeks Dagny T

Recommend 3 Share Sort by Newest

Join the discussion…

kaushik Lele • 5 months ago

To visually understand the solution see YouTube video explanation
https://www.youtube.com/wat...

Reply • Share ›

Abhishek • 8 months ago

1) Make HashSet or Set for All the substring for string 1
2) Now again produce all the substring for string 2 if above set contains the sub string than update value of max

```
import java.util.*;

public class HelloWorld{

static void longestSubstring(String s1,String s2){

Set<string> set = new HashSet<string>();
int max = 0;
int maxIndex=-1;

for(int i = 1 ; i < s1.length();++i){
for(int j = 0 ; j+i <= s1.length();j++){
set.add(s1.substring(j,j+i));
}
}

for(int i = 1 ; i < s2.length();++i){
for(int j = 0 ; j+i <= s2.length();j++){
if(set.contains(s2.substring(j,j+i))){
if(max<i){ max="i;" maxindex="j;" }="" }="" }="" }="" system.out.println("max="" length=""
"+="" max);="" system.out.println("max="" index="" "+="" maxindex);="" }="" public=""
static="" void="" main(string="" args[]){="" longestsubstring("abccb","bcabc");="" }=""
}="">
```

Reply • Share ›

Jingguo Yao • 8 months ago

"substrings" should be "prefixes" in the following text:

> The idea is to find length of the longest common suffix for all substrings of both strings and store these lengths in a table.

Reply • Share ›

bhavik gujarati • 9 months ago

To print Longest Common Substring:

http://ideone.com/jfwM2p

1 Reply • Share ›

Sally Talks • 10 months ago

I think following definition is correct : -
LCSuff(X, Y, m, n) = LCSuff(X, Y, m-1, n-1) + 1 if X[m] = Y[n]

LCSuff(X, Y, m-1, n-1) Otherwise (if X[m]] != Y[n])

Reply • Share ›

Vishal Vaibhav ➚ Sally Talks • 10 months ago

consider it 0-indexed

Reply • Share ›

Apurv Shah • a year ago

Hi, Is there any other method? What to do when string length is too large, for example what if both strings length is 100000? It's not possible to allocate memory for such a large size array (i.e LCSuff 2D array in this example).

Reply • Share ›

Poompatai Puntidpong ➚ Apurv Shah • a year ago

Hi,Apurv
After you understand the algorithm,you will see that you only need access to 2 line at a time(i-th row and i-1-th row) so you only need two arrays at a time and switch use them using modulo 2 . You will need just 2*min(m.size,n.size) memory but time complexity still o(mn).

Reply • Share ›

Rencan Fang • a year ago

A little tip: we'd better use the vector other than int array, int array may lead to constant value problem

Reply • Share ›

Billionaire • 2 years ago

Please distinguish with Longest Common Subsequence(LCS) problem!

http://www.geeksforgeeks.or...

1 ^ | v • Reply • Share ›

vaibhav29 Mod ➚ Billionaire • a year ago

Concepts of a substring and a subsequence are different. Subsequence is a generalization of substring.

Please read this article : http://www.geeksforgeeks.or...

^ | v • Reply • Share ›

Ashwin Iyengar • 2 years ago

why do we assign LCS[i][j] = 0 if x[i]! = y[j] , in such cases if we have string abcde & zbcf then wont the entry for zbc & abcd be 0 , shouldnt it be 2? Please explain

^ | v • Reply • Share ›

sevensevens ➚ Ashwin Iyengar • a year ago

try to write out the LCS matrix then you will see it's 2

^ | v • Reply • Share ›

Neeraj Dhelariya B Tech. Dept • 2 years ago

C : Easier way for those who don't like dynamic programming

see this link: http://code.geeksforgeeks.o...

O(m*n)

int LCSubStr(char X[],char Y[], int m, int n){

int length=0;

for(int i=0; i<m; i++){="" int="" prev_count="0,count=0;" for(int="" j="0;" j<n;="" j++){="" if(x[i+j]="=Y[j])" {="" count++;="" if(j="=n-1)" prev_count="max(prev_count,count);" }="" else{prev_count="max(prev_count,count);" count="0;" }="" }="" length="max(length," prev_count);="" }="" return="" length;="" }="">

^ | v • Reply • Share ›

This comment was deleted.

Tequila ➚ Guest • 2 years ago

your topdown soln does not have time complexity O(n*m)

^ | v • Reply • Share ›

Mr. Lazy ➚ Tequila • 2 years ago

You can check the number of function calls for solving unsolved subproblems never go beyond n*m....

Reply • Share ›

Tequila ➚ Mr. Lazy • 2 years ago

number of subproblems are no doubt O(n*m)
but your func getsub () can take worst case O(min(n,m)) time so it will be O(n*m*min(n,m))

Reply • Share ›

Mr. Lazy ➚ Tequila • 2 years ago

Yes my bad, you are right! Deleted!

Reply • Share ›

Andy • 2 years ago

This is incredibly inefficient. And with proper practice can be completed in no more then 5-7 lines of c++11(and up)

Reply • Share ›

tinku kasyap ➚ Andy • 2 years ago

do share ur code

Reply • Share ›

Billionaire • 2 years ago

The meaning of LCSubStr(X, Y, m, n) should be longest common substring ending with m and n in X, Y separately.

Reply • Share ›

Ashwin Iyengar • 2 years ago

Can anyone explain why we build table for m+1 n+1 instead of m n?

Reply • Share ›

lucy ➚ Ashwin Iyengar • 2 years ago

because we first fill zero in first col and first row and our ans in lcs[m][n].And index start from zero so total length will be m+1,n+1

Reply • Share ›

Manoj Saini • 2 years ago

A Simple Dynamic solution. Which returns the length of LCS and the index of last character in First String.

public class LongestCS {

public class LongestCS {

public static void main(String s[]){

char X[] = "OldSite:GeeksforGeeks.org".toCharArray();// first String

char Y[] = "NSite:GeeksQuiz.com".toCharArray();// second string

int lx = X.length;

int ly = Y.length;

int max = Integer.MIN_VALUE;

int endIndex = 0;

int D[][] = new int[lx][ly];

for(int i=0;i<lx;i++){ for(int="" j="0;j<ly;j++){" if(i="=0" ||="" j="=0){" if(x[i]="=Y[j]){" d[i][j]="1;" if(max<d[i][j]){="" max="D[i][j];" endindex="i;" }="" }="" }else{="" if(x[i]="=Y[j]){" d[i][j]="D[i-1][j-1]+1;" if(max<d[i][j])="" {="" max="D[i][j];" endindex="i;" }="" }="" }="" }="" }="" system.out.println(max);="" system.out.println(endindex);="" }="" }="">

Cracker • 2 years ago

@GeeksforGeeks: My solution does not take O(n^2) space. It can be done using O(n) space only. See below
http://algods-cracker.blogs...

lucy • 2 years ago

@GeeksforGeeks can we solve it KMP?

prashant jha • 2 years ago

http://ideone.com/jjCWQ0

Mission Peace • 2 years ago

https://www.youtube.com/wat... Watch my video for LCS explanation

ganga ➚ Mission Peace • 7 months ago

This is one of the best and easily understood solution explanation seen so far for LCS problem! Thanks a lot!:)

Saksham Sharma ➚ Mission Peace • 9 months ago

Thanks a ton =)

^ | v • Reply • Share ›

bang • 2 years ago

how about this? O(mn) and constant space complexity

string findLCS(string s1, string s2) {
pair<int,string>res;
int maxLen = 0 ;

for(int i=0; i<s1.length() ;="" ++i)="" {="" for(int="" j="0;" i<s2.length()="" ;="" ++j)="" {="" if(s1.at(i)="=" s2.at(j))="" {="" int="" l="i," m="j," len="0;" string="" commonstr="" ;="" while(s1.at(l)="=" s2.at(m))="" {="" commonstr="" +="s1.at(l);" l++;="" m++;="" len++;="" }="" if(res.size()="" <="0" ||="" res.first="" <="" len)="" {="" res="new" pair(len,commonstr);="" }="" j="" +="len;" }="" }="" }="" }="">

^ | v • Reply • Share ›

Monu Kesarwani • 2 years ago

solution for longest common substring length without using extra space in 0(m*n) time complexity.

#include<stdio.h>
#include<string.h>
#define MAX 20
int lon_substring(char *s1, int len1, char *s2, int len2)
{
int i,j,k,count=0,max_count=0,i_start,i_end;
for(i=0;i<len1;i++) {="" k="i;" for(j="0;j<len2;)" {="" if(s1[k]="=s2[j])" {="" while(s1[k]="=s2[j]" &&="" j<len2)="" {="" count++;="" k++;="" j++;="" }="" if(max_count<count)="" {="" max_count="count;" }="" count="0;" }="" else="" {="" j++;="" k="i;" }="" }="" }="" return="" max_count;="" }="" int="" main()="" {="" char="" s1[max]="GeeksforGeeks" ;="" char="" s2[max]="Geeks" ;="" int="" l1="strlen(s1);" int="" l2="strlen(s2);" printf("longest="" substring="" length="%d",lon_substring(s1,l1,s2,l2));" return="" 0;="" }="">

^ | v • Reply • Share ›

Tongtong Flora Liu • 3 years ago

I dont think that's right.
It's for sub sequence, not subString!

^ | v • Reply • Share ›

Anurag Singh ➚ Tongtong Flora Liu • 3 years ago

No. It's for longest common substring.
Longest Common Subsequence is at
Dynamic Programming | Set 4 (Longest Common Subsequence)

^ | v • Reply • Share ›

typing.. • 3 years ago

a quality solution!!!!!
geeksforgeeks hats off..... :)

1 ^ | v • Reply • Share ›

Rohit Sharma • 3 years ago

//PRINT SUB-STRING !!

```
#include<stdio.h>

#include<string.h>

int longComSubStr(char *,char *);

//int max(int,int);

int main()

{

char str1[100],str2[100];

printf("\nEnter the string1 : \n");

gets(str1);

printf("\nEnter the string2 : \n");
```

see more

^ | v • Reply • Share ›

winter ➚ Rohit Sharma • 3 years ago

your output is wrong, you just save the last index which is matched and print backward?

^ | v • Reply • Share ›

winter ➚ winter • 3 years ago

just save one index_i (or index_j) now you have result to take output array of size result. and save in that.
Here is the modified later part.

```
int k= result;

char output[result]; //have a out put of length result

output[result] = '\0';

while(k>0)

{

output[k-1] = str1[index_i-1];

index_i--;

k--;

}

printf("\nThe sub string : %s ",output);
```

Reply • Share ›

AlienOnEarth • 3 years ago

@GeeksforGeeks:

I think the recurrence relation is not written properly. It says max of Max(LCSuff(X, Y, i, j), ?) and what else? There is only 1 parameter for Max(a,b)

Reply • Share ›

GeeksforGeeks Mod ➚ AlienOnEarth • 3 years ago

Please take a closer look. Here i and j vary from 1 to n. It is maximum for all possible values of i and j. Let us know if this clarifies.

1 Reply • Share ›

Anjaneya Alluri • 3 years ago

Hi, Can anyone provide the Memoization version of this in Java

Reply • Share ›

sawan • 3 years ago

The Problem can be solved with O(m*n) time and O(m) space complexity.
In the below program the result array is basically result[2], to store the start and end index of resultant substring.

```
void LCS(char str1[],char str2[], int result[]) {

int len1=strlen(str1),len2=strlen(str2);
```

char temp[2][len2];

int z=0;

for(int i=0;i<len1;i++){ for(int="" j="0;j<len2;j++){" if(str1[i]="=str2[j]){" if(i="=0" ||="" j="=0)" temp[i%2][j]="1;" else="" temp[i%2][j]="temp[(i%2+1)%2][j-1]+1;" if(temp[i%2][j]="">=z){

z=temp[i%2][j];

result[0]=i-z+1;

result[1]=i;

see more

1 · Reply · Share ›

Ronak Hingar • 3 years ago

Here is the code in Java

public class LCS {

public String LongestCommonSubsequence(String x, String y, int i, int j) {

String s = "";

if (i == x.length() || j == y.length()) {

return "";

}

if (x.charAt(i) == y.charAt(j)) {

s += x.charAt(i) + LongestCommonSubsequence(x, y, i + 1, j + 1);

return s;

} else {

see more

2 · Reply · Share ›

Anjaneya Alluri ➚ Ronak Hingar • 3 years ago

isn’t creating so many strings in the process of finding substring a bad idea ? Can this be optimized so that we don’t create so many strings.

· Reply · Share ›

prashant • 3 years ago

```
int fun(char st1[],char st2[],int low1,int low2)

{

int max=0;

if((low1>=strlen(st1))||(low2>=strlen(st2)))

return max;

for(int i=low1;i<strlen(st1);i++) {="" int="" k="search(st1[i],st2,low2);" if(k!="-1)" {="" int="" p="1+fun(st1,st2,i+1,k+1);" if(p="">max)

max=p;

}

}

return max;

}
```

Reply • Share ›

jeremy • 3 years ago

excellent job!

Reply • Share ›

Guest • 3 years ago

```
/*find the length of Longest Common Substring */

#include<iostream>

#include<string>

using namespace std;

string s="OldSite:GeeksforGeeks.org";

string str="NewSite:GeeksQuiz.com";

int main()

{

int i,index=0,max_size=0,count=0,l,k;

for( i=0; i<s.length(); i++)="" {="" for(int="" j="0;" j<str.length();="" j++)="" {="" 
```

if(s[i]="=str[j])" {="" count="0;" l="i;" k="j;" while(s[l]="=str[k]" &&="" l="">=0 && k>=0)

see more

Reply • Share ›

Guest • 3 years ago

/*find the length of Longest Common Substring */
#include<iostream>
#include<string>
using namespace std;

string s="OldSite:GeeksforGeeks.org";
string str="NewSite:GeeksQuiz.com";

int main()
{
int i,index=0,max_size=0,count=0,l,k;
for(i=0; i<s.length(); i++)="" {="" for(int="" j="0;" j<str.length();="" j++)="" {=""
if(s[i]="=str[j])" {="" count="0;" l="i;" k="j;" while(s[l]="=str[k]" &&="" l="">=0 && k>=0)
{
count+=1;
l--;
k--;
}

see more

3 Reply • Share ›

eragon • 3 years ago

fails on input harry sally answer is 2 for ay. Code gives 1

1 Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Add Disqus Add Privacy DISQUS