GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Dynamic Programming | Set 29 (Longest Common Substring)

Given two strings 'X' and 'Y', find the length of the longest common substring. For example, if the given strings are "GeeksforGeeks" and "GeeksQuiz", the output should be 5 as longest common substring is "Geeks"

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 C++ implementation of the above solution.

```
/* Dynamic Programming solution to find length of the longest common substrin
#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;
}
```

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

Related Topics:

- Recursively print all sentences that can be formed from list of word lists
- Check if a given sequence of moves for a robot is circular or not
- Find the longest substring with k unique characters in a given string
- Function to find Number of customers who could not get a computer
- Find maximum depth of nested parenthesis in a string
- Find all distinct palindromic sub-strings of a given string
- Find if a given string can be represented from a substring by iterating the substring “n” times
- Suffix Tree Application 6 – Longest Palindromic Substring

Tags: Dynamic Programming

Writing code in comment? Please use ideone.com and share the link here.

Join the discussion…

Ashwin Iyengar • 3 days ago

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

• Reply • Share ›

lucy ➚ Ashwin Iyengar • 3 hours 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 • 8 days ago

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

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);="" }="" }="">

• Reply • Share ›

Cracker • 22 days 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.blogspot...

• Reply • Share ›

lucy • a month ago

@GeeksforGeeks can we solve it KMP?

• Reply • Share ›

prashant jha • a month ago

http://ideone.com/jjCWQ0

• Reply • Share ›

Mission Peace • 2 months ago

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

• Reply • Share ›

bang • 6 months 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;" }="" }="" }="" }="">

• Reply • Share ›

Monu Kesarwani • 6 months 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;="" }="">

• Reply • Share ›

Tongtong Flora Liu • 7 months ago

Tongtong Flora Liu • 7 months ago

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

• Reply • Share ›

Anurag Singh ➔ Tongtong Flora Liu • 7 months ago

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

• Reply • Share ›

typing.. • 8 months ago

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

1 • Reply • Share ›

Rohit Sharma • 10 months 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

• Reply • Share ›

winter ➔ Rohit Sharma • 9 months ago

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

• Reply • Share ›

winter winter • 9 months 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 • a year 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 • a year 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 • a year ago

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

Reply • Share ›

sawan • a year 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&amp;lt;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 ^ | v • Reply • Share ›

Ronak Hingar • a year 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 ^ | v • Reply • Share ›

Anjaneya Alluri ➦ Ronak Hingar • a year 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 • a year 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 • a year ago

excellent job!

• Reply • Share ›

Guest • a year 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 • a year 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 • a year ago

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

1 • Reply • Share ›

Guest ➚ eragon • a year ago

ay is a subsequence.. we are looking for a substring

1 • Reply • Share ›

jaaaames ➚ eragon • a year ago

the code is for longest contiguous string

• Reply • Share ›

wgpshashank • a year ago

Please correct wht you have written There will be O(m^2) substrings actually there will be

n(n+1)/2 exact sub-strings for string of length n.

• Reply • Share ›

Kartik ➚ wgpshashank • a year ago

Nothing seems to be wrong to me. For a string of length m, there are always O(m^2) possible substrings.

• Reply • Share ›

dharmendra • 2 years ago

Nice explanation, I really liked it.

There is another variant of this question where the interviewer asks you to find the length of the longest common subsequence instead of substring.

I wrote an article for the same, if anybody came here searching for the topic.
http://techieme.in/techieme/dy...

• Reply • Share ›

AlienOnEarth ➚ dharmendra • a year ago

LCS is already present on GeeksforGeeks

• Reply • Share ›

learner • 2 years ago

In case someone is a java guy :)

```
[sourcecode language="JAVA"]
public class LCM {
public static void main(String[] args) {
System.out.println(lcm(args[0],args[1]));
}

static String lcm(String s1, String s2){
int result = 0;
int[][] lcmdp = new int[s1.length()+1][s2.length()+1];
int index=0;

for(int i=0;i<s1.length();i++){
for(int j=0;j<s2.length();j++){
if(s1.charAt(i) == s2.charAt(j)){
lcmdp[i+1][j+1] = lcmdp[i][j] + 1;
//result = Math.max(lcmdp[i+1][j+1], result);
if(result < lcmdp[i+1][j+1]){
```

see more

• Reply • Share ›

Vikas Gupta • 2 years ago

Munish first take two string and find the LCS of both string and let name it as Max_LCS. Now take the next string and find LCS of this string and Max_LCS and update the Max_LCS. This way proceed down to n strings and the final Max_LCS will be the answer..Hope u got it..:)

• Reply • Share ›

Munish Singla • 2 years ago

When more then two strings are given, how to find the longest common substring in all of them...

2 • Reply • Share ›

gargsanjay • 2 years ago

we can do this through hash
cresate hashtable of size(small string*chars)
store indexes of occurence of every character in first string
now pick elements of 2nd string one by one and and try to match them with every index present of that character in hash table.
O(m*n)

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

Vignesh Miriyala • 2 years ago

Santosh Kumar the string is not sorted to do binary search ;).

• Reply • Share ›

Santosh Kumar • 2 years ago

for(i=0;i<str2.lenght;i++){

int index=searchBinary(str1, str2[i]);.
while(str1[index++]=str[j++]).
{.
result+ = str1[index];.
}.
if(result.length()>finalLCM.length()).
finalLCM=result;.
}

time complexcity: O(n*lcm*logn).

• Reply • Share ›

firoz • 2 years ago

```
#include<stdio.h>
```

```
void main()
{
  int i,j,k,n=0,m,t;
  char ar[50],arr[50];
  clrscr();
  printf("enter first string:\t");
  gets(ar);
  printf("\nenter Second String:\t");
  gets(arr);
  for(i=0;ar[i]!='&#92&#48';i++)
     {
      for(j=0,k=i;arr[j]!='&#92&#48';j++)
        {
          if(ar[k]==arr[j])
            {
                k++; t=k;
             }
```

see more

• Reply • Share ›

tushar ➦ firoz • 2 years ago

Nice one , algo do not uses any extra memory and working perfectly.

• Reply • Share ›

madhur ➦ tushar • 2 years ago

it is not working for all cases.

• Reply • Share ›

Born Actor • 2 years ago

```
 //if a[i][j]=0 when s1[i]!=s2[j], we get the largest contiguous subsequence from the code
// if we remove the commented part and comment the lines from 52 to 59, we get the largest


#include <iostream>
#include<string>
#include<sstream>
#include<iomanip>
# include <stdio.h>
# include <math.h>
#include <vector>
#include <stdlib.h>
using namespace std;
int m;
```

```
int n;
int a[50][50];
int max(int a,int b, int c);
int function();
```

see more

• Reply • Share ›

anonymous • 2 years ago

Code for printing the substring as well

```
#include<stdio.h>
#include<string.h>

void lcsub(char x[],char y[],int m,int n)
{

        int i,j;
        int table[m+1][n+1];

        int max=0,ind=0;
        for(i=0;i<=m;i++)
          for(j=0;j<=n;j++)
            {
                if(i==0 || j==0)
                  table[i][j]=0;
```

see more

2 • Reply • Share ›

Balpreet Singh anonymous • 2 years ago

The code for printing string is not correct, it will not print complete string.

Correct code is

printf("maximum LCSUBSTRING IS :");
for(i=ind-max;i<ind;i++) printf("%c",x[i]);="" printf("\n");="">

• Reply • Share ›

PG • 2 years ago

You guys are doing awesome work. It really helps a lot. Keep up the good work forever :)

```
/* Paste your code here (You may delete these lines if not writing code) */
```

Waqas Khan • 2 years ago

can u please tell me code in c.

caijinlong • 2 years ago

```
 #include <iostream>
#include <vector>
using namespace std;

int location = 0;

int LCSubstring(char *x, char *y, const int m,const int n)
{
        //int LCSbuff[m + 1][n + 1];
        //int **LCSbuff = new int*[m + 1];
        vector<vector<int> > LCSbuff;
        //int* a[4];
        LCSbuff.resize(m + 1);

        int result = 0;
        for (int i = 0; i <= m; i++)
        {
                LCSbuff[i].resize(n + 1);
```

see more

caijinlong • 2 years ago

#include
#include
using namespace std;

int location = 0;

int LCSubstring(char *x, char *y, const int m,const int n)
{
//int LCSbuff[m + 1][n + 1];
//int **LCSbuff = new int*[m + 1];
vector<vector > LCSbuff;
//int* a[4];
LCSbuff.resize(m + 1);

int result = 0:

```
for (int i = 0; i <= m; i++)
{
LCSbuff[i].resize(n + 1);
//LCSbuff[i] = new int [n + 1];
```

see more

• Reply • Share ›

Subhajit • 2 years ago

Solution which prints the substring also:

```
 #include<iostream>
#include<string.h>

using namespace std;
int max(int a,int b)
{
        if(a>b)
                return a;
        else
                return b;
}
int main()
{
        char *s="forGeeksGeeks";
        char *s1="QuizGeeks";
```

see more

• Reply • Share ›

caijinlong ➚ Subhajit • 2 years ago

```
#include
#include
using namespace std;

int location = 0;

int LCSubstring(char *x, char *y, const int m,const int n)
{
//int LCSbuff[m + 1][n + 1];
//int **LCSbuff = new int*[m + 1];
vector<vector > LCSbuff;
//int* a[4];
LCSbuff.resize(m + 1);
```

```
int result = 0;
for (int i = 0; i <= m; i++)
{
LCSbuff[i].resize(n + 1);
//LCSbuff[i] = new int [n + 1];
```

see more

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

- Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 2 hours ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 2 hours ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 2 hours ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago