The Learning Point

Search this site

Main Page: The Complete Index at a Glance | Mathematics | Computer Science | Physics | Electrical Science and Engineering | An Introduction to Graphics and Solid Modelling | Test Preparations | Ace the Programming Interviews | CoursePlex: Online Open Classes from Coursera, Udacity, etc | About

Computer Science >

Algorithms: Dynamic Programming - Longest Common Sub-sequence with C Program Source Code

Quick Links to Various Sections on the Main Page

The Learning Point
Computer Science
Algorithms: An introduction to Dynamic Programming
CS - Data Structures and Algorithms
CS - Programming Interviews
CS - Miscellaneous C Programs
CS - Intro to DB Queries
Electrical Science and Engineering
Electrical Sci - DC Circuits
Electrical Sci - Digital Electronics
Mathematics
Mathematics - Linear Algebra
Mathematics - Geometry
Mathematics - Single Variable Calculus
Mathematics - Game Theory
Mathematics - Operations Research
Physics
Physics - Basic Mechanics
Physics - Electrostatics
Sitemap
Recent site activity

These are Quick Links to sections of the Main Page

0

To go through the C program / source-code, scroll down to Traduci ***of this page***

Longest Common Subsequence

A subsequence of a given sequence is the given sequence with just some elements left out (order should be from left-to-right, not necessarily consecutive).. A common sequence of two sequences X and Y, is a subsequence of both X and Y.A longest common subsequence is the one with maximum length. For example, if X = {A,B,C,B,D,A,B} and Y = { B,D,C,A,B,A} then the longest common subsequence is of length 4 and they are {B,C,B,A} and {B,D,A,B}.

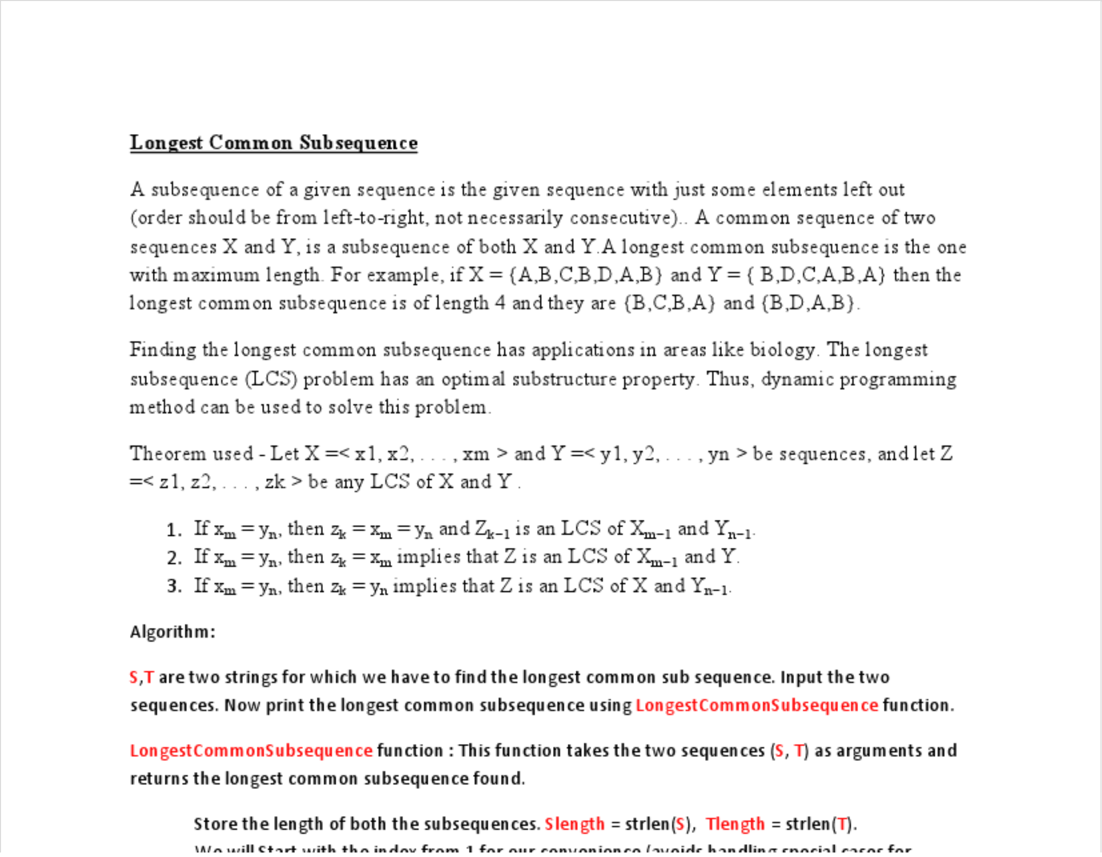
Finding the longest common subsequence has applications in areas like biology. The longest subsequence (LCS) problem has an optimal substructure property. Thus, the dynamic programming method can be used to solve this problem.

Theorem used - Let X =< x1, x2, . . . , xm > and Y =< y1, y2, . . . , yn > be sequences, and let Z =< z1, z2, . . . , zk > be any LCS of X and Y .

1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Zk−1 is an LCS of X_{m-1} and Y_{n-1}.
2. If $x_m = y_n$, then $z_k = x_m$ implies that Z is an LCS of X_{m-1} and Y.
3. If $x_m = y_n$, then $z_k = y_n$ implies that Z is an LCS of X and Y_{n-1}.

Complete Tutorial with Examples:

1 / 3



Longest Common Subsequence

A subsequence of a given sequence is the given sequence with just some elements left out (order should be from left-to-right, not necessarily consecutive).. A common sequence of two sequences X and Y, is a subsequence of both X and Y.A longest common subsequence is the one with maximum length. For example, if X = {A,B,C,B,D,A,B} and Y = { B,D,C,A,B,A} then the longest common subsequence is of length 4 and they are {B,C,B,A} and {B,D,A,B}.

Finding the longest common subsequence has applications in areas like biology. The longest subsequence (LCS) problem has an optimal substructure property. Thus, dynamic programming method can be used to solve this problem.

Theorem used - Let X =< x1, x2, . . . , xm > and Y =< y1, y2, . . . , yn > be sequences, and let Z =< z1, z2, . . . , zk > be any LCS of X and Y .

1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.
2. If $x_m = y_n$, then $z_k = x_m$ implies that Z is an LCS of X_{m-1} and Y.
3. If $x_m = y_n$, then $z_k = y_n$ implies that Z is an LCS of X and Y_{n-1}.

Algorithm:

S,T are two strings for which we have to find the longest common sub sequence. Input the two sequences. Now print the longest common subsequence using LongestCommonSubsequence function.

LongestCommonSubsequence function : This function takes the two sequences (S, T) as arguments and returns the longest common subsequence found.

Store the length of both the subsequences. Slength = strlen(S), Tlength = strlen(T).

We will Start with the index from 1 for our convenience (avoids handling special cases for

C Program Source Code for the Longest Common Subsequence problem

```
#include<stdio.h>
#include<string.h>
#define maxn 100100
int max(int a,int b)
{
        return a>b?a:b;
}
int LongestCommonSubsequence(char S[],char T[])
{
        int Slength = strlen(S);
        int Tlength = strlen(T);
        /* Starting the index from 1 for our convinience (avoids handling special cases for negative indices) */
        int iter,jter;
        for(iter=Slength;iter>=1;iter--)
        {
                S[iter] = S[iter-1];
        }
        for(iter=Tlength;iter>=1;iter--)
        {
                T[iter] = T[iter-1];
        }
        int common[Slength+1][Tlength+1];
        /* common[i][j] represents length of the longest common sequence in S[1..i], T[1..j] */
        /* Recurrence:  common[i][j] = common[i-1][j-1] + 1 if S[i]==T[j]
                                     = max(common[i-1][j],common[i][j-1]) otherwise
        */
        /*common[0][i]=0, for all i because there are no characters from string S*/
        for(iter=0;iter<=Tlength;iter++)
        {
                common[0][iter]=0;
        }
        /*common[i][0]=0, for all i because there are no characters from string T*/
        for(iter=0;iter<=Slength;iter++)
        {
                common[iter][0]=0;
        }
        for(iter=1;iter<=Slength;iter++)
        {
                for(jter=1;jter<=Tlength;jter++)
                {
                        if(S[iter] == T[jter] )
                        {
                                common[iter][jter] = common[iter-1][jter-1] + 1;
                        }
                        else
                        {
                                common[iter][jter] = max(common[iter][jter-1],common[iter-1][jter]);
                        }

                }
        }
        return common[Slength][Tlength];

}
int main()
{
        char S[maxn],T[maxn];/* S,T are two strings for which we have to find the longest common sub sequence. */
        scanf("%s%s",S,T);
        printf("%d\n",LongestCommonSubsequence(S,T));

}
```

Rough notes about the Algorithm implemented in the code above:

S,T are two strings for which we have to find the longest common sub sequence. Input the two sequences. Now print the

LongestCommonSubsequence function : This function takes the two sequences (S, T) as arguments and returns the

Store the length of both the subsequences. Slength = strlen(S), Tlength = strlen(T). We will Start with the index from
Declare **common[Slength][Tlength]**. Where, common[i][j] represents length of the longest common sequence in S[1..i], T[1
If there are no characters from string S, common[0][i]=0 for all i or if there are no characters from string T, common
Recurrence: for i=1 to Slength
for j=1 to Tlength
common[i][j] = common[i-1][j-1] + 1, if S[i]=T[j]. Else, common[i][j] = max(common[i-1][j],common[i
variables as arguments and returns the maximum of them.

Return **common[Slength][Tlength]**.

Related Tutorials (common examples of Dynamic Programming):

Integer Knapsack problem	An elementary problem, often used to introduce the concept of dynamic programming.
Matrix Chain Multiplication	Given a long chain of matrices of various sizes, how do you parenthesize them for the purpose of multiplication - how do you chose which ones to start multiplying first?
Longest Common Subsequence	Given two strings, find the longest common sub sequence between them.

Some Important Data Structures and Algorithms, at a glance:

Arrays : Popular			

Sorting and Searching Algorithms			
Bubble Sort	Insertion Sort	Selection Sort	Shell Sort
Merge Sort	Quick Sort	Heap Sort	Binary Search Algorithm
Basic Data Structures and Operations on them			
Stacks	Queues	Single Linked List	Double Linked List
Circular Linked List	1.		

Tree Data Structures			
Binary Search Trees	Heaps	Height Balanced Trees	
Graphs and Graph Algorithms			
Depth First Search	Breadth First Search	Minimum Spanning Trees: Kruskal Algorithm	Minumum Spanning Trees: Prim's Algorithm
Dijkstra Algorithm for Shortest Paths	Floyd Warshall Algorithm for Shortest Paths	Bellman Ford Algorithm	
Popular Algorithms in Dynamic Programming			
Dynamic Programming	Integer Knapsack problem	Matrix Chain Multiplication	Longest Common Subsequence
Greedy Algorithms			
Elementary cases : Fractional Knapsack Problem, Task Scheduling	Data Compression using Huffman Trees		

Like Send 10 people like this. Sign Up to see what your friends

Recommend this on Google

Add a comment...

Om Bachchan · Amity School of Engineering & Technology

The input to this problem is a pair of strings of letters A = a1... am possible. The rules are as follows:

- For a cost of 3 you can delete any letter.
- For a cost of 4 you can insert a letter in any position.
- For a cost of 5 you can replace any letter by any other letter.

For example, you can convert A = abcabc to B = abacab via the fol which at cost of 3 can be converted to ababc, which at cost of 3 ca abacb, which at cost of 4 can be converted to abacab. Thus the tot the cheapest possible conversion.

I/O description. Input: the two strings on separate consecutive lin.

Reply · 1 · Like · October 24 at 7:48am

Facebook social plugin

Accedi | Attività recente del sito | Segnala abuso | Stampa pagina | Rimuovi accesso | Powered by Google Sites