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

Printing Longest Common Subsequence

Given two sequences, print the longest subsequence present in both of them.

Examples:
LCS for input Sequences “ABCDGH” and “AEDFHR” is “ADH” of length 3.
LCS for input Sequences “AGGTAB” and “GXTXAYB” is “GTAB” of length 4.

We have discussed Longest Common Subsequence (LCS) problem in a previous post. The function discussed there was mainly to find the length of LCS. To find length of LCS, a 2D table L[][] was constructed. In this post, the function to construct and print LCS is discussed.

Following is detailed algorithm to print the LCS. It uses the same 2D table L[][].

1) Construct L[m+1][n+1] using the steps discussed in previous post.

2) The value L[m][n] contains length of LCS. Create a character array lcs[] of length equal to the length of lcs plus 1 (one extra to store \0).

2) Traverse the 2D array starting from L[m][n]. Do following for every cell L[i][j]
…..**a)** If characters (in X and Y) corresponding to L[i][j] are same (Or X[i-1] == Y[j-1]), then include this character as part of LCS.
…..**b)** Else compare values of L[i-1][j] and L[i][j-1] and go in direction of greater value.

The following table (taken from Wiki) shows steps (highlighted) followed by the above algorithm.

		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
		Ø	**M**	**Z**	**J**	**A**	**W**	**X**	**U**
0	**Ø**	**0**	0	0	0	0	0	0	0

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

1	**X**	0	0	0	0	0	0	1	1
2	**M**	0	**1**	1	1	1	1	1	1
3	**J**	0	1	1	**2**	2	2	2	2
4	**Y**	0	1	1	2	2	2	2	2
5	**A**	0	1	1	2	**3**	3	3	3
6	**U**	0	1	1	2	3	3	3	**4**
7	**Z**	0	1	2	2	3	3	3	4

Following is C++ implementation of above approach.

```
/* Dynamic Programming implementation of LCS problem */
#include<iostream>
#include<cstring>
#include<cstdlib>
using namespace std;

/* Returns length of LCS for X[0..m-1], Y[0..n-1] */
void lcs( char *X, char *Y, int m, int n )
{
   int L[m+1][n+1];

   /* Following steps build L[m+1][n+1] in bottom up fashion. Note
      that L[i][j] contains length of LCS of X[0..i-1] and Y[0..j-1] *
   for (int i=0; i<=m; i++)
   {
     for (int j=0; j<=n; j++)
     {
       if (i == 0 || j == 0)
         L[i][j] = 0;
       else if (X[i-1] == Y[j-1])
         L[i][j] = L[i-1][j-1] + 1;
       else
         L[i][j] = max(L[i-1][j], L[i][j-1]);
     }
   }

   // Following code is used to print LCS
   int index = L[m][n];

   // Create a character array to store the lcs string
```

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

```
   char lcs[index+1];
   lcs[index] = '\0'; // Set the terminating character

   // Start from the right-most-bottom-most corner and
   // one by one store characters in lcs[]
   int i = m, j = n;
   while (i > 0 && j > 0)
   {
      // If current character in X[] and Y are same, then
      // current character is part of LCS
      if (X[i-1] == Y[j-1])
      {
          lcs[index-1] = X[i-1]; // Put current character in result
          i--; j--; index--;     // reduce values of i, j and index
      }

      // If not same, then find the larger of two and
      // go in the direction of larger value
      else if (L[i-1][j] > L[i][j-1])
         i--;
      else
         j--;
   }

   // Print the lcs
   cout << "LCS of " << X << " and " << Y << " is " << lcs;
}

/* Driver program to test above function */
int main()
{
  char X[] = "AGGTAB";
  char Y[] = "GXTXAYB";
  int m = strlen(X);
  int n = strlen(Y);
  lcs(X, Y, m, n);
  return 0;
}
```

Output:

```
LCS of AGGTAB and GXTXAYB is GTAB
```

References:

http://en.wikipedia.org/wiki/Longest_common_subsequence_problem

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

Related Tpoics:

- Suffix Array | Set 2 (nLogn Algorithm)
- Rearrange a string so that all same characters become d distance away
- Recursively remove all adjacent duplicates
- Find the first non-repeating character from a stream of characters
- Dynamic Programming | Set 33 (Find if a string is interleaved of two other strings)
- Remove “b” and “ac” from a given string
- Dynamic Programming | Set 29 (Longest Common Substring)
- Write your own atoi()

15 Tweet 3 0

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

6 Comments **GeeksforGeeks**

Sort by Newest

695 Subscribe

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 13 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 33 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 33 minutes ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

Join the discussion…

Vijay Nair • 11 hours ago

i have a feeling you dont need to do the diagnol thing at all,

instead i tried this and it works for all cases that i tried

```
char *pString = new char[SizeofString+1];
int maxSeen = 0;
int writeCtr =0;
for (int j = 1; j <= n; j++)
{
if (L[m][j] > maxSeen)
{
pString[writeCtr++] = second[j-1];
maxSeen = L[m][j];
}
}

pString[writeCtr]='\0';
```

this works just fine and its order n which is less than the mXn it would take to

^ | ⌄ • Reply • Share ›

GOPI GOPINATH • 5 days ago

anyone...give me the applications where we use LCS...and i request geeksfog articles describing few applications where we can use the concept...

^ | ⌄ • Reply • Share ›

Suman ➦ GOPI GOPINATH • 2 days ago

File diff uses LCS

^ | ⌄ • Reply • Share ›

GOPI GOPINATH ➦ Suman • 2 days ago

differentiating files ??

^ | ⌄ • Reply • Share ›

kaushik Lele • 7 days ago

I have modified above code a little. So that in first loop of i-j itself the characters
soon as loop is over, LCS is ready

```
public class LCSLongestCommonSubsequence {

public static void main(String[] args) {
String str1 = "AGGTAB";
String str2 = "GXTXAYB";
int m = str1.length();
int n = str2.length();
System.out.println("Length of LCS is n" + lcs( str1, str2, m, n ) );
}

public static Result lcs( String x, String y, int m, int n )
{
Result L[][] = new Result[m+1][n+1];
int i, j;
/* Following steps build L[m+1][n+1] in bottom up fashion. Note
that L[i][j] contains length of LCS of X[0..i-1] and Y[0..j-1] */
```

see more

^ | ⌄ • Reply • Share ›

kaushik Lele ➦ kaushik Lele • 7 days ago

Code is shared at http://ideone.com/LJ1JDV

^ | ˅ • Reply • Share ›

Subscribe Add Disqus to your site