

| **Title: Implementation of Longest Common Subsequence String Matching Algorithm** |
| --- |

C:\Users\kjsce\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{5FDD96F3-22C8-4D59-868F-90D5F9DB251F}\{60C3D34C-3195-44D0-AA17-6AB634A64984}\ResourceMap\{4AF8A3AA-EF41-4A04-9E5B-8EBA3E1FB62C}

**Objective:**  To compute longest common subsequence for the given two strings.C:\Users\kjsce\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{5FDD96F3-22C8-4D59-868F-90D5F9DB251F}\{60C3D34C-3195-44D0-AA17-6AB634A64984}\ResourceMap\{FEEB4CA7-5BD6-421D-ADE1-E96E7076DCD8}

**CO to be achieved:**

| CO 2 | Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies. |
| --- | --- |
| CO 3 | Analyze and solve problems for   different string matching algorithms. |

C:\Users\kjsce\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{5FDD96F3-22C8-4D59-868F-90D5F9DB251F}\{60C3D34C-3195-44D0-AA17-6AB634A64984}\ResourceMap\{09422953-361B-4CD7-919E-F480B17AB785}

**Books/ Journals/ Websites referred:**

**1.                  Ellis horowitz, Sarataj Sahni, S.Rajsekaran,” Fundamentals of computer algorithm”, University Press**

**2.                  T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein,” Introduction to algortihtms”,2nd Edition ,MIT press/McGraw Hill,2001**

**3.                  http://www.math.utah.edu/~alfeld/queens/queens.**

C:\Users\kjsce\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{5FDD96F3-22C8-4D59-868F-90D5F9DB251F}\{60C3D34C-3195-44D0-AA17-6AB634A64984}\ResourceMap\{FEEB4CA7-5BD6-421D-ADE1-E96E7076DCD8}

**Pre Lab/ Prior Concepts:**

Data structures, Concepts of algorithm analysis

C:\Users\kjsce\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{5FDD96F3-22C8-4D59-868F-90D5F9DB251F}\{60C3D34C-3195-44D0-AA17-6AB634A64984}\ResourceMap\{4AF8A3AA-EF41-4A04-9E5B-8EBA3E1FB62C}

**Historical Profile:**

C:\Users\kjsce\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{5FDD96F3-22C8-4D59-868F-90D5F9DB251F}\{60C3D34C-3195-44D0-AA17-6AB634A64984}\ResourceMap\{560338E6-1E3B-490C-94D2-A0996917E746}Given 2 sequences, *X* = *x*1 *, ..., xm*  and *Y* = *y*1 *, ... , yn* ,   find a subsequence common to both whose length is longest.  A subsequence doesn’t have to be consecutive, but it has to be in order.

**New Concepts to be learned:**

String matching algorithm, Dynamic programming approach for LCS, Applications of LCS.

Recursive **Formulation:**

Define *c*[*i, j* ] = length of LCS of *Xi* and *Y j* .

Final answer will be computed with *c*[*m, n*].

c[i, j]= 0

if i=0 or j=0.

c[i, j]= c[i − 1, j − 1] + 1

if i,j>0 and xi=yj

c[i, j]= max(c[i − 1, j ], c[i, j − 1])

if i, j > 0 and xi <> yj

**Algorithm: Longest Common Subsequence**

**Compute length of optimal solution-**

**LCS-LENGTH** *( X , Y, m, n)*

**for** *i* ← 1 **to** *m*

**do** *c*[*i,* 0] ← 0

**for** *j* ← 0 **to** *n*

**do** *c*[0*, j* ] ← 0

**for** *i* ← 1 **to** *m*

**do for** *j* ← 1 **to** *n*

**do if** *xi* = *y j*

**then** *c*[*i, j* ] ← *c*[*i* − 1*, j* − 1] + 1

*b*[*i, j* ] ← “≈”

**else  if** *c*[*i* − 1*, j* ] ≥ *c*[*i, j* − 1]

**then** *c*[*i, j* ] ← *c*[*i* − 1*, j* ]

*b*[*i, j* ] ← “↑”

**else** *c*[*i, j* ] ← *c*[*i, j* − 1]

*b*[*i, j* ] ← “←”

**return** *c* and *b*

**Print the solution-**

**PRINT-LCS*(b, X , i, j )***

**if** *i* = 0 or *j* = 0

**then return**

**if** *b*[*i, j* ] = “≈”

**then** PRINT-LCS*(b, X , i* − 1*, j* − 1*)*

print *xi*

**elseif** *b*[*i, j* ] = “↑”

**then** PRINT-LCS*(b, X , i* − 1*, j )*

**else** PRINT-LCS*(b, X , i, j* − 1*)*

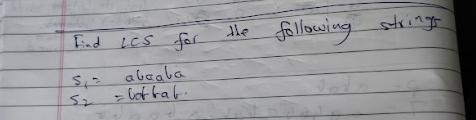
Initial call is PRINT-LCS*(b, X , m, n)*.

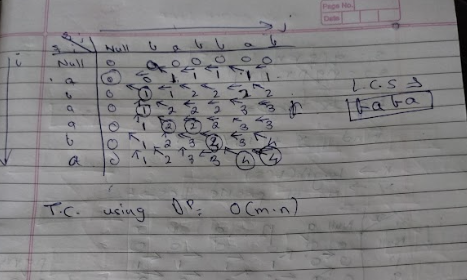
*b*[*i, j* ] points to table entry whose subproblem we used in solving LCS of *Xi*

and *Y j.*

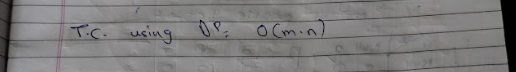
When *b*[*i, j* ] = ≈, we have extended LCS by one character. So longest com- mon subsequence = entries with ≈ in them.

**Example: LCS computation**

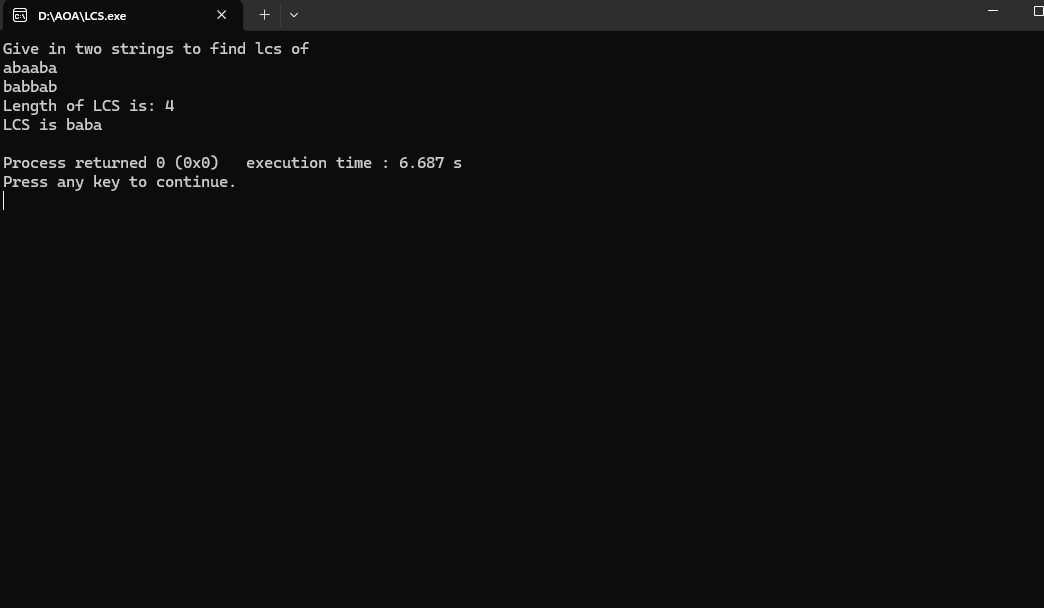
****

****

**Analysis of LCS computation**

****

**Output:**



**Algorithm:**

 #include <bits/stdc++.h>

using namespace std;

int main()

{

cout << "Give in two strings to find lcs of\n";

string a, b;

cin >> a >> b;

int m = a.size();

int n = b.size();

vector<vector<int>> v(m + 1, vector<int>(n+1));

for(int i = 1; i <= m; i++)

{

for(int j = 1; j <= n; j++)

{

if(a[i-1] == b[j-1])

{

v[i][j] = v[i-1][j-1]+1;

}

else

{

v[i][j] = max(v[i-1][j], v[i][j-1]);

}

}

}

int i = m, j = n;

string lcs = "";

while(i >= 0 && j >= 0)

{

if(a[i] == b[j])

{

lcs = a[i] + lcs;

i--;

j--;

}

else if(v[i-1][j] < v[i][j-1])

{

i--;

}

else

{

j--;

}

}

cout << "Length of LCS is: " << v[m][n] << "\n";

cout << "LCS is " << lcs << "\n";

}

**CONCLUSION:**

We have computed LCS of the two strings and analysed the algorithm.