**Batch: D2 Roll No.: 16010122323**

**Experiment No. 10**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

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



**Objective:** To compute longest common subsequence for the given two strings.

**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. |



**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.**



**Pre Lab/ Prior Concepts:**

Data structures, Concepts of algorithm analysis



**Historical Profile:**

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.

**Code of LCS computation**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void LCS\_LENGTH(char X[], char Y[], int m, int n, int c[m+1][n+1], char b[m+1][n+1]) {

for (int i = 0; i <= m; i++)

c[i][0] = 0;

for (int j = 0; j <= n; j++)

c[0][j] = 0;

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

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

if (X[i-1] == Y[j-1]) {

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

b[i][j] = '=';

} else if (c[i-1][j] >= c[i][j-1]) {

c[i][j] = c[i-1][j];

b[i][j] = 'u';

} else {

c[i][j] = c[i][j-1];

b[i][j] = 'l';

}

}

}

printf("Length of Longest Common Subsequence: %d\n", c[m][n]);

printf("Longest Common Subsequence: ");

char lcs[c[m][n]];

int i = m, j = n, k=0;

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

if (b[i][j] == '=') {

lcs[k++] = X[i-1];

i--;

j--;

} else if (b[i][j] == 'u') {

i--;

} else {

j--;

}

}

while(k > 0)

printf("%c", lcs[--k]);

}

int main() {

char X[] = "aggtab";

char Y[] = "gxtxayb";

int m = strlen(X);

int n = strlen(Y);

int c[m+1][n+1];

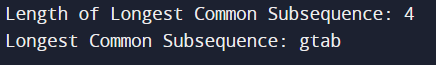
char b[m+1][n+1];

LCS\_LENGTH(X, Y, m, n, c, b);

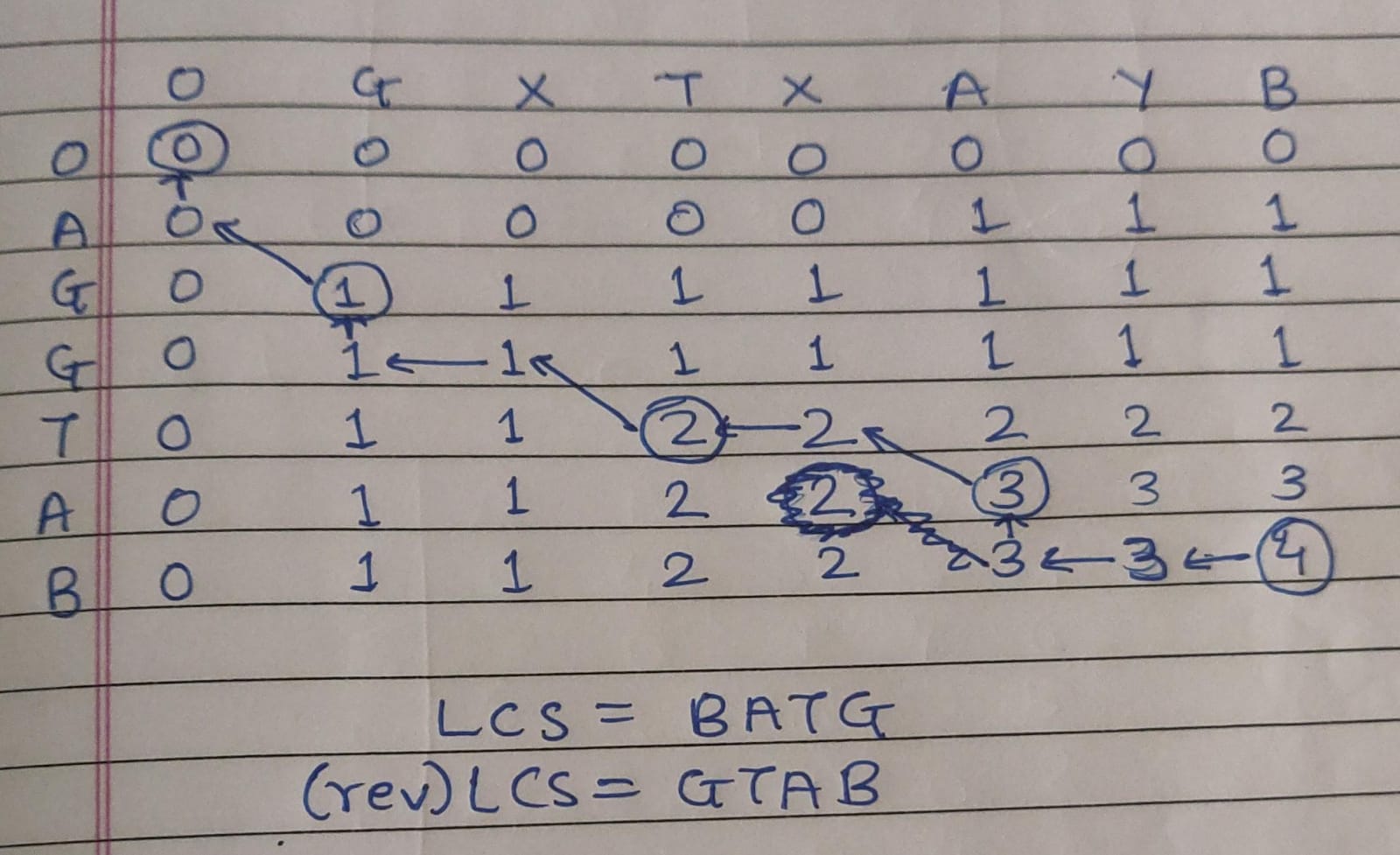
return 0;

}

**Output:**



**Example: LCS computation**

****

**Analysis of Algorithm:**

Time Complexity: O(m \* n)

Auxiliary Space: O(m)

**CONCLUSION:**

Learned to compute longest common subsequence for the given two strings.