

Name	Aditi Nilesh Bhutada
UID no.	2021700009
Experiment No.	4

AIM:	Experiment based on divide and conquer approach. Finding the running time of an algorithm. The understanding of running time of algorithms is explored by implementing two basic sorting algorithms namely merge sort and quick sort.
PROGRAM	
THEORY:	<p>The longest common subsequence (LCS) is defined as the longest subsequence that is common to all the given sequences, provided that the elements of the subsequence are not required to occupy consecutive positions within the original sequences.</p> <p>If S_1 and S_2 are the two given sequences then, Z is the common subsequence of S_1 and S_2 if Z is a subsequence of both S_1 and S_2. Furthermore, Z must be a strictly increasing sequence of the indices of both S_1 and S_2.</p>
INPUT:	Input two strings either of numbers or alphabets
ALGORITHM:	<p>In dynamic programming approach we store the values of longest common subsequence in a two dimensional array which reduces the time complexity to $O(n * m)$ where n and m are the lengths of the strings. Let the input sequences be X and Y of lengths m and n respectively. And let $dp[n][m]$ be the length of LCS of the two sequences X and Y.</p> <p>We iterate through a two dimensional loop of lengths n and m and use the following algorithm to update the table $dp[][]$:-</p> <ul style="list-style-type: none"> • If any of the loop variable i or j is 0, then $dp[i][j] = 0$. • if $X[i-1] = Y[j-1]$, i.e., when the characters at ith and jth index matches, $dp[i][j] = 1 + dp[i-1][j-1]$.

	<ul style="list-style-type: none"> Otherwise, store the maximum value we get after considering either the character X[i] or the character Y[j], i.e., $dp[i][j] = \max(dp[i][j-1], dp[i-1][j])$. <p>To retrieve the subsequence, follow a bottom-up approach. When the length of subsequence considering the Y[j] is greater than that considering X[i], decrement j, increment i when the length of subsequence considering the Y[j] is less than that considering X[i]. Whenever the length of subsequence considering the Y[j] is equal to that considering X[i], include the character in the answer.</p>
PROGRAM:	<pre> #include<stdio.h> #include<string.h> int i,j,m,n,c[20][20]; char x[20],y[20],b[20][20]; int max(int a, int b); void print(int i,int j) { if(i==0 j==0) return; if(b[i][j]=='c') { print(i-1,j-1); printf("%c",x[i-1]); } else if(b[i][j]=='u') print(i-1,j); else print(i,j-1); } void lcs() { m=strlen(x); n=strlen(y); for(i=0;i<=m;i++){ c[i][0]=0; </pre>

```

for(i=0;i<=n;i++){
    c[0][i]=0;

    //c, u and l denotes cross, upward and downward directions
    respectively
    for(i=1;i<=m;i++){
        for(j=1;j<=n;j++){
            {
                if(x[i-1]==y[j-1])
                {
                    c[i][j]=c[i-1][j-1]+1;
                    b[i][j]='c';
                }
                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';
                }
            }
        }
    }
}

int max(int a, int b)
{
    if(a>b){
        return a;
    }
    else
        return b;
}

// Returns length of LCS for X[0..m-1], Y[0..n-1]
int lcs_length(char* x, char* y,int m,int n)

```

```

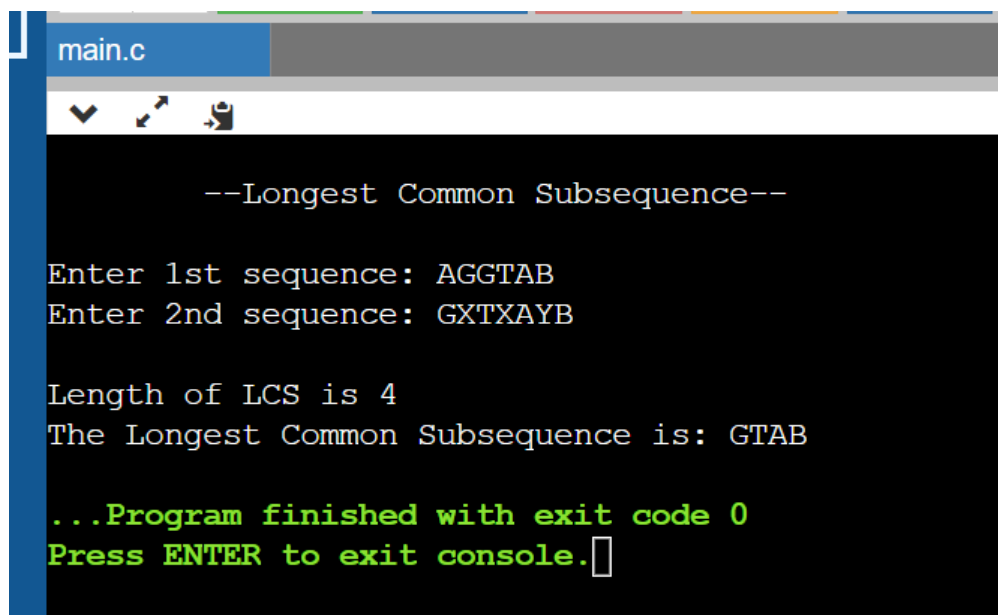
{
    if (m == 0 || n == 0)
        return 0;
    if (x[m - 1] == y[n - 1])
        return 1 + lcs_length(x, y, m - 1, n - 1);
    else
        return max(lcs_length(x, y, m, n - 1), lcs_length(x, y, m - 1, n));
}

int main()
{
    printf("\n\t--Longest Common Subsequence--\n");
    printf("\nEnter 1st sequence: ");
    scanf("%s", x);
    printf("Enter 2nd sequence: ");
    scanf("%s", y);
    printf("\nLength of LCS is %d", lcs_length(x, y, m, n));
    printf("\nThe Longest Common Subsequence is: ");
    lcs();
    print(m, n);

    return 0;
}

```

OUTPUT:



The screenshot shows a terminal window with the following output:

```

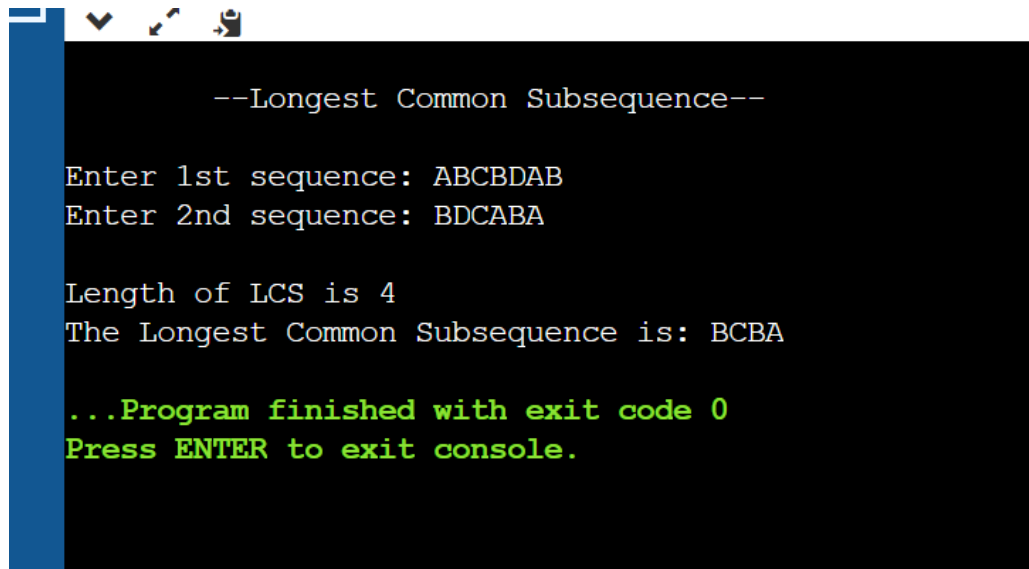
--Longest Common Subsequence--

Enter 1st sequence: AGGTAB
Enter 2nd sequence: GXTXAYB

Length of LCS is 4
The Longest Common Subsequence is: GTAB

...Program finished with exit code 0
Press ENTER to exit console.

```



```
--Longest Common Subsequence--

Enter 1st sequence: ABCBDAB
Enter 2nd sequence: BDCABA

Length of LCS is 4
The Longest Common Subsequence is: BCBA

...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT ANALYSIS:	<p>We have presented two approaches to find the longest common subsequences-</p> <ul style="list-style-type: none">• Naive Recursive approach in $O(2^n)$ time-complexity.• Dynamic Programming approach in $O(n * m)$ time-complexity.
CONCLUSION:	<p>In this experiment I understood about how to find the longest common subsequence of two given sequences using dynamic programming</p>